Ulrich Frank, Stefan Strecker

Open Reference Models

Community-driven Collaboration to Promote Development and Dissemination of Reference Models

Reference models constitute a reification of a promising vision: Higher quality of information systems at less cost through reuse of confirmed domain knowledge and systems design. Paradoxically, however, development and, in particular, reuse of reference models has been rather limited both in practice and academia. In this paper, we develop the notion of open reference models based on analogies to free and open source software development. We show how "openness" of reference models affects their development and use, and outline strategic options for a first open reference modelling initiative. Our findings suggest that community-driven collaborative modelling projects resolve the current paradox of reference model research and practice.

Earlier versions appeared as [FrSK07a] and [FrSK07b]. Preliminary considerations are provided in [KoSF06].

1 The reference modelling paradox

Reference models are conceptual models associated with the claim to represent abstractions that suit not only one enterprise but fit the needs of a defined class of organisations; typically an industry or economic sector, e.g., [Sche94]. They embody two pivotal claims: On the one hand, they are intended to provide *descriptions* of an application domain that fit actual requirements. On the other hand, reference models are aimed at delivering *blueprints* for a distinctively good design of information systems that accounts for potentials offered by advanced technologies. Hence, their underlying claim is descriptive *and* prescriptive at the same time. From an economic point of view, they promise the realisation of an attractive vision: better software at less cost. Software based on reference models does not only benefit from a thoroughly developed design, but also from a higher level of integration: Software integration requires individual parts to communicate, which in turn implies the need for common concepts. A reference model provides a common *semantic reference* system to the information systems built upon it. At the same time, reference models promise a tremendous decline in software development costs through massive reuse. On the one hand, this is attributed to the multiple reuse of a reference model itself. On the other hand, reuse relates to the software artefacts built according to a reference model. The reuse of the latter is also fostered by the reference models' contribution to integration. If, for example, enterprise resource planning (ERP) systems are built on such a reference system, chances are that modules from third-party vendors integrate relatively seamlessly with a packaged solution – provided they comply to the reference model as well. Taken to its extreme, this argument suggests the development of interconnected reference models for all industries and sectors worldwide. The concepts of those models would serve as the lingua franca of electronic business – allowing for exchanging high level business objects and for composing tightly integrated cross-organisational business processes. In addition to conceptual models of software systems, e.g., data and object models, enterprise models also include representations of the surrounding action system, e.g., business process models and models of organisational resources. Hence, reference enterprise models provide a blueprint not only for the design of software systems, but also for organising the firm and for aligning information systems with business [Fran07].

Reference models do not only promise to improve quality and economics of information systems development, they also offer a very attractive perspective for scientific research. Scientific knowledge in its ideal form is presented as a theory. While there is no common notion of a theory, it can be regarded as a linguistic construction consisting of concepts and propositions (hypotheses), which serve to describe the invariant phenomena of a certain domain. Therefore, concepts and propositions of a theory ought to be general (or lawful) in the sense that they apply to classes of phenomena, not just to one particular occurrence, and that they are valid not just for a certain period of time, but at best forever. Similar to theories, the descriptive part of reference models is intended to capture general features of information systems and the action systems they are embedded in. Different from theories, the prescriptive part of reference models informs about opportunities beyond current practice – and, hence, corresponds to the claim of applied research, namely, to support practice through the dissemination of advanced concepts.

The construction of comprehensive reference models is an intellectually challenging and demanding task. Due to their partly prescriptive nature, they do not only include abstractions of existing but also of possible future worlds. Therefore, they support the claim of the Information Systems (IS) discipline to provide an orientation for the design of advanced information systems and corresponding action systems. Hence, reference models can be regarded as object and objectification of IS research, e.g., [BeSc04; FeLo04]—representing ideal vehicles for disseminating scientific knowledge. For more than two decades, reference models and reference modelling have been researched [ScWi06; BrBu06]. However, the development, dissemination, and use of reference models remained far behind even modest expectations. A recent survey identifies 30 models of which eight were developed outside of academia [FeLo03; FeLZ05]. Only few reference models are known to a wider audience and those are by-products of ERP systems [Rose03], e.g., the SAP reference models [KeTe98] based on work by Scheer [Sche94] and others [ScNü00]. Even these preliminary findings indicate that the development of reference models, their diffusion, and use are impeded by serious obstacles.

Complexity is an essential characteristic of developing reference models. Often, reference models consist of hundreds of concepts and relationships among them. Think, for example, of a reference model that serves as a conceptual foundation of an ERP systems, e.g., [CuKL98]. The development of abstractions suited to satisfy a multitude of organisations will usually require a comprehensive analysis of a number of organisations.

> *Hypothesis 1:* The efforts required for the construction of reference models will usually go beyond the capabilities of a single researcher or even a single research group.

> *Hypothesis 2:* The current academic system does not provide an adequate reward system for the development of reference models. Academic careers depend crucially on journal publications. However, because of their sheer size, reference models are hardly suited for publication in journals or conference proceedings.

As far as the success of a reference model is concerned, its actual use is of crucial importance. It requires prospective users to be convinced of the benefits provided by a reference model. That means that there is need to build trust in the quality and usability of reference models. The conviction – and probability – that a reference model fits actual needs will increase if potential users are involved in the development process.

> *Hypothesis 3:* A business firm will be very reluctant to participate in a research project on developing a reference model, because the return on investment is hard to predict.

Apparently, reference modelling research and practice face a seeming paradox: On the one hand, reference models hold promising prospects for both scientific research and business practice. On the other hand, current organisational conditions and incentive structures inhibit realising these promises. To resolve the paradox, (1) an effective organisation of joint projects including researchers and representatives of prospective users, and (2) effective incentives both for researchers and for participating business firms respectively their employees are needed. Restrictive licensing of reference models has in the past impeded innovative forms of organisations and motivation [FeLZ05]. The recent "open licensing" movement has, however, prepared the ground for new forms of collaboration based on less restrictive legal provisions. One of the prime examples of community-driven collaboration is free and open source software[1]: FOSS promotes collaboration of developers and users, has led to artefacts of surprising quality, and, in some cases, to widespread use of artefacts.

Inspired by the apparent successes of some FOSS projects, this paper investigates whether FOSS development provides a suitable orientation for resolving the reference model paradox (see [Broc04] for additional considerations). The next section reviews liter-

[1] The differences between "free software" [Stal99; Stal02] and "open source software" [Pere99] do not affect our analysis. Hence, we consciously include both approaches in the analysis, where we refer to them as free/open source software or FOSS for short.

ature to summarise key characteristics of FOSS development. Based on the analysis, a comparison of the two artefacts, i.e., source code and conceptual models, is carried out in Sec. 3 to investigate to what extent these characteristics apply to reference models. Section 4 investigates effects of “openness” of reference models on their development and use. In Sec. 5, we outline strategic options to initialise a first community-driven reference modelling initiative. The final section reports on first steps undertaken to establish open modelling projects.

2 The free/open source phenomenon as a source for inspiration

Free and open source software marks one of the most astonishing developments in recent information technology history. With roots in academic communities, the idea of FOSS has evolved into a technical, economic, and societal phenomenon [LeTi02]: Complex software systems are being developed collaboratively by online communities of geographically dispersed participants. Systems software such as GNU/Linux, Apache, and MySQL as well as end-user applications such as Mozilla Firefox and OpenOffice have become widely known examples of FOSS. With lesser known projects such as GNU Enterprise, tinyERP, and Compiére, FOSS now also includes enterprise-wide information systems [DKRW05].

The FOSS phenomenon releases substantial yet previously unrealised creativity and productivity potentials; fosters collaboration beyond institutional and professional boundaries; and produces artefacts of surprising quality, acceptance, and use that provide a remarkable challenge for even large software corporations. At the same time, the communities creating the artefacts and the artefacts themselves are surrounded by an aura of independence, freedom, and quality awareness. They have received remarkable recognition and sympathy from both professional and academic users. There is, however, no reason for elation: The few overly successful FOSS projects are opposed by a far larger number of initiatives that fail [HeSc03]. A fundamental question therefore pertains to what features distinguish those online communities that have led to the development of widely used, i.e., successful artefacts [CrHA06].

The characteristic features of FOSS and its development are documented in reports of involved 'insiders', e.g., [MoFH02; Raym99; RaTr99], and in studies of observing 'outsiders', e.g., [Fitz06; Ghos05; StGo06]. A scholarly account of the phenomenon is, however, confronted with a number of obstacles: Idealised and programmatic self-portrayals obstruct a clear view and detract from actual manifestations [Raym99; Stal02]. Undue generalisations from single instances convey the impression that FOSS is a singular and homogeneous phenomenon [Raym99]. However, FOSS research reveals a rather complex and heterogeneous phenomenon [Ross04]. Unfortunately, only few studies account explicitly for the multi-facetted interactions and interdependencies in FOSS projects (see [Aspe05; MoSp00; Tuom05] for first attempts). At present, literature on FOSS, thus, provides only early indications as to which factors contribute to establishing a community and to developing a useable artefact [Kris02]. The following characterisation synthesises findings on four aspects, i.e., licensing, organisation, coordination, and incentives from available accounts of the phenomenon.

Constituent are licensing terms with which the licensee is provided with access to the source code as well as non-exclusive rights to use, modify, and distribute the software without royalties [LeTi05]. A FOSS project is driven by a community of – at least partially volunteer – software developers who rely on end users' input and feedback [CrHo04]. The community shares norms, values, and beliefs that serve as a foundation of the project culture and contribute to a common identity [StGo06]. Due to the spatial distribution of participants, communication and collaboration is networked using asynchronous, e.g., mailing lists, as well as synchronous, e.g., chat systems, electronic media [Coff06]. The organisation is specific to a project but typically evolves with the changing requirements of the community [Gall01] and with the artefacts created [YNYK04].

FOSS projects feature a high degree of division of labour with task assignment largely based on self-selection [Benk02], i.e., individuals identify tasks that fit their own schedule. This organisational model entails the need for coordination. Coordination occurs through signalling of individuals or groups of individuals to the community. According to [Gall01], coordination in FOSS projects is largely based on self-control and social control through the community. Typical role concepts [CrHo04], power distributions, and decision structures [YNYK04] have evolved as means of coordination in FOSS projects.

The primary incentive scheme in FOSS projects is based on individual reputation [MaMA00]. Individuals gain recognition by peers through significant contributions to the project. Serious contributions are generally peer-reviewed by the community. Acquired reputation is honoured through preferred positions in the organisational hierarchy and serves to achieve a respectable social status in a FOSS community. This reputation-based incentive scheme motivates contributions from newcomers as well as from active community members. Newcomers strive to build a reputation. Already active contributors intend to maintain their reputation. The scheme also promotes timely submission of quality contributions. The potential negative consequences connected to a loss of reputation function as an additional incentive [ShSR02]. Besides incentives directed at extrinsic motivation, numerous empirical studies show a range of intrinsic individual motives to participate in FOSS projects including having fun in creating high quality software and the personal need for the software, e.g., [BoRo03; HeNH03].

Although the factors decisive for the success of FOSS projects are still unknown [CrHA06], the brief analysis of present literature warrants first theses:

- The success of FOSS projects is founded, among other things, on an underlying ideology that is shown to positively affect mutual trust and quality of communication and, lastly, project success [Gall01; StGo06].
- Empirical findings suggest high extrinsic and intrinsic motivation of participants [HaOu02; HeNH03; Lakh05], which, in turn, is likely to lead to a distinct focus on the project's goals and, thereby, contributes to project success.
- The modularisation of software artefacts increases the potential for division of labour and parallelisation of work, e.g., with respect to software tests and bug fixing [Raym99], and reduces the need for coordination [BoRo03].
- Division of labour through self-selection fosters work on tasks that demand skills other than coding, e.g., documentation and end user support, because participants are able to contribute individual competencies.
- Self-control of participants and social control through the community, respectively, lower the demand for (explicit) coordination [Gall01]. Reputation functions as an effective means of self-control; peer review works as an effective instrument of social control [MaMA00].
- Experiential rules emphasise the involvement of all stakeholders including prospective end users from the very beginning of a project as well as early and frequent releases [Raym99]. Both seem to increase effectiveness and efficiency of the development process.

3 The relation between “open source” and “open models"

In analogy to FOSS, we refer to an “open reference model” (or “open model” for short) as a (reference) model licensed under terms which provide the licensee with unrestricted access to all model representations and documentations as well royalty-free, non-exclusive rights to use, copy, modify, and (re-)distribute the model and its documentation. Apart from these fundamental legal aspects [KoSF06], we assert further commonalities. Source code and reference models are both:

- digitally represented artefacts accessible to modularisation and, hence, to a development based on division of labour. They are both suited, in principle, for collaborative development by communities of geographically dispersed participants.
- purposeful and sophisticated artefacts whose development depends on design decision that require extensive technical knowledge.
- artefacts that satisfy a formal syntax and, thus, are accessible to automated analyses and transformations.
- artefacts that demand associated documentation to enable and facilitate maintenance and future enhancements.
- artefacts whose quality cannot easily be measured, e.g., considering their aesthetics. Therefore, they both require suitable processes of quality assurance.
- artefacts with a (potentially) large economic impact with respect to both royalty-free use and to new business models which accompany this use.

There are, however, a number of more or less apparent differences:

- Reference models as compared to source code largely abstract from implementation-level constraints and emphasise the consideration of the respective domain specific terminology.
- Software rewards the developer with a satisfying experience of having created an executable artefact that immediately supports a certain task relevant to its user. Reference models are, normally, not executable. Their use requires further adaptations, transformations, and interpretations.

- Source code is intended for machine processing first and human comprehensibility second. Reference models, on the other hand, inherently require interpretation on part of the user in the light of the framed (modelling) purpose and model documentation. Assessing a model's quality is therefore more difficult than evaluating the quality of source code.
- Programming skills are more widely available than conceptual modelling skills, especially with respect to reference modelling which is probably restricted to a comparatively smaller group of modellers.
- FOSS development profits from the myth still surrounding software development. Reference modelling is not accompanied by such legends and myths.
- FOSS also profits from the recognition 'free' and 'open source' software receive from society at large. It is widely perceived as an act of emancipation from the omnipotence of international corporations. A similar effect could accompany the development of reference models whose development and use has in the past been controlled by large software vendors. Different from software, however, vendor dominance on reference models is not as obvious and, therefore, more difficult to convey.

Against this background, we now turn to the question to what extent the key characteristics of FOSS can (presumably) be transferred to open models. The following theses summarise our assessment.

Norms, values, beliefs: The FOSS ideology is largely independent from source code as artefact and, therefore, transferable to open reference modelling.

Reputation as incentive mechanism: Reference modelling is an intellectually challenging task. It is to be expected that significant contributions will be recognised by the community and increase the reputation of the contributor. The remarkable reputation, the so called “amigos” (Booch, Rumbaugh and Jacobson) gained for their contribution to the specification of the UML supports this assumption. However, considering the time and effort required to construct reference models, it is highly questionable whether a special social status within the community provides sufficient incentives to contribute. It seems very likely that additional provisions are necessary to foster acquired reputation beyond the open model community, especially in the academic modelling community.

Peer review: A thorough evaluation of a reference model demands significant time and effort as well as superb modelling skills. Reviewing open reference models therefore requires additional incentives as well. This seems especially important with respect to reviews by domain experts from industry.

Experiential rules: Many of the rules sketched by Raymond and others appear conferrable to open modelling, e.g., “release early, release often” [Raym99].

Involving individual competencies: Besides modelling skills, open model projects require additional skills, primarily domain-specific knowledge but also skills with regard to documentation, user support etc. Again, additional stakeholder-specific incentives are needed to motivate contributions in these areas, but especially considering active participation by practitioners.

Developer and user participation: Involving both developers and users in the open model development process seems possible given adequate access. Achieving a comparable degree of parallelisation of work largely depends on the achievable degree of modularisation of models and of modelling tasks.

Modularity: Integrated meta-models provide a way, in principle, to modularise reference models. However, it remains a substantial challenge for open model projects to break down one reference model into several workable parts which later can be (re-)integrated into a consistent whole.

4 Effects of “openness” on the development and use of reference models

Reference models can be regarded as common, shared languages for communication not only among actors in the systems development process but also among software artefacts built upon it. A language which is publicly accessible, royalty-free, and individually adaptable provides its users with a greater independence than products marketed for similar use. Independence protects investments into learning, applying, and possibly altering the language in the long run, and, hence, provides advantages over proprietary models.

Considering the present difficulties in locating and accessing reference models [FeLZ05], it is likely that repositories similar to Sourceforge[2] and other FOSS portals will emerge which simplify searching, locating and accessing models and their documentation. It is not unlikely that the facilitated access to reference models changes patterns of their use. Similar to FOSS, the sheer availability of open reference models

[2] http://www.sourceforge.org

is also likely to change users' expectations towards the availability of *any* conceptual model, which, in turn, increases pressure on vendors to license their proprietary models under accepted open model terms. Apart from increasing the number of available open models, participation of software vendors in the open model community realises synergies leading to a higher quality of models which, in turn, is expected to affect their use.

Open access to reference models is also expected to promote their use in teaching and training. Reference models provide a laboratory for learning, e.g., about information systems development in an application domain, because they convey a solid conceptual foundation of information systems and surrounding action systems – on a level of abstraction suited for academic contemplation. Their use in teaching and training is likely to affect their use in application contexts: If learners, e.g., university students, are familiar with a reference model, it is likely that they are going to apply it in later occupations. This argument is supported by cost considerations as well: If employees are familiar with a certain model, it reduces the cost of use and maintenance in a corporate context.

Participation in the development of open reference models offers several benefits to prospective model users: Individual contributions are likely to increase the usability and usefulness of a reference model. Reviewing a model helps to identify problems from the user's perspective to benefit from respective solutions in future releases. Prospective users profit from community support and, indirectly, from the artefacts created. Especially small and medium-sized companies are expected to benefit from access to (scholarly) modelling know-how and respective resources. Given a critical mass of participants, it is expected that individual contributions are offset by the community's contributions at large and, thus, create incentives to contribute at the individual level.

Participation in the development of open reference models also promises benefits for researchers: Open reference models provide an effective medium for exchange with practitioners and, in particular, with prospective users, which is especially important considering that reference models only display their ascribed benefits if used in many practical applications. Researchers gain access to the problems and challenges faced by practitioners and profit from their domain knowledge. If prospective users participate by submitting change requests to the community, collaboration with practitioners in open model communities provides (empirical) access to user requirements and their variance—connected to the challenging task to cover the variance with appropriate conceptual abstractions.

Model "openness" entails community-driven model development involving prospective users from business and academia. The user involvement improves the acceptance and, lastly, the use of artefacts in both research and teaching. It also helps to reduce the known barrier to adopt models in practice that have been created in academia. Community-driven modelling also permits to deal with tasks too demanding for individuals through the bundling of distributed resources.

Lastly, the proximity to FOSS projects fosters mutual exchange with the FOSS community, e.g., with projects developing ERP systems. Although the developers of these systems recognise the need for a conceptual foundation [Mcco99; Wils99], we are not aware of any project that currently focuses on building one. The benefits are likely mutual: The FOSS community can benefit significantly from the availability of open models, while the input from FOSS projects feeds into the initiative [KoSF06].

Our considerations suggest that essential aspects of FOSS development carry over – in principle – to the construction of conceptual models and that our conception of "open models" promises to overcome the paradoxical situation in reference modelling. However, it becomes apparent that the first open model initiative is faced with severe challenges, e.g., with respect to creating incentives for participation. The next section discusses options for initialising a first open model initiative.

5 Initialising first open model projects

Obviously, a missionary call for open model projects is not sufficient. Rather, an appropriate strategy is needed to induce a critical mass of modellers, software developers, and end users to establish sustainable efforts. Such a strategy pertains to the selection of promising modelling domains, modelling purposes, levels of abstraction, and an associated process of initialisation. The following hypotheses are confined to those aspects that appear most relevant to the initialisation of first open models projects. Each hypothesis is stated under a "ceteris paribus" assumption. We distinguish two groups of actors, researchers and firms, both of which may take on the role of developer and user of a reference model. A further analysis is provided in [FrSK07b].

Modelling domain: To enable economies of scale, the chosen modelling domain should be of importance to a large number of potential users who, at the same time, should exhibit a high degree of commonalities and overlapping requirements.

Modelling purposes: Several complementary purposes are conceivable:

Alternative 1) The development of reference models as a foundation for software development is especially attractive, if software development profits from the use of reference models in a convincing manner, e.g., with respect to integration, software reuse, and competitive capabilities.

Alternative 2) The development of reference models as a foundation for communication between (existing) systems, e.g., as a specification for programming interfaces (in the sense of loosely coupled systems) is especially attractive, if either the respective domain has to deal with legacy systems which cannot be replaced in due time or if inter-organisational value chains are modelled.

Alternative 3) The development of reference models as an orientation for structuring complex action systems, e.g., business processes is especially attractive, if competitive pressure is high in the domain and the domain is undergoing structural change at the same time.

Actors: It is assumed that the development of open reference models benefits from the fact that developers are also prospective model users. *For firms*, participation in the development of open models is attractive if

- the reference model promises to improve the competitiveness of the firm, especially with regard to overpowering competitors;
- the reference model reduces market entry barriers;
- the reference model serves as a medium for additional services;
- the participation promises a valuable qualification of participating employees, in part due to the exchange with academic institutions;
- the divulgation of firm-specific advantages is overcompensated by the expected benefits from using the reference model.

For academic researchers, participation in the development of open models is attractive if

- the modelling tasks involve intellectual challenges and is clearly related to discipline-specific research tasks;
- the involvement promises differentiated access to the users' practices;
- the reference model itself serves as a research subject, e.g., to develop transformation methods, and modelling tools;
- the reference model promises to enhance teaching.

Levels of abstractions: Reference models can be used for different levels of abstraction. On the one hand, this refers to abstraction from specific particularities of concrete instances, i.e., to how generic a model is. On the other hand, it refers to the difference between meta models used for (modelling) language specification, and models as applications of modelling languages. There are reasons to choose either strategy. To quickly achieve a critical mass of participants, timely development of artefacts is especially important, which recommends the use of well-known modelling languages.

Genericness: On the one hand, the more a reference model abstracts from the specifics of single instances, i.e., the further removed from the application it is, the more intellectually demanding its development which is attractive from a scientific point of view. Also, the potential for reuse and integration is increased. On the other hand, the closer a reference model stays to applications, the higher its immediate benefits for those whose requirements are met. This also increases the chances of realisation. Relevant decision criteria include feasibility, (empirical) variance, and available resources. It seems recommendable to start with a bottom-up approach to quickly produce first results, and then add a top-down approach.

Model vs. meta-model: On the one hand, developing modelling languages acts as a deterrent to firms because it puts their prior investments in modelling languages, e.g., UML, at danger and it also delays the development of applicable reference models. On the other hand, the development of modelling languages is an especially demanding task and is a well-recognised research area in software engineering and IS research. Relevant decision criteria include the quality of existing modelling languages and modelling tools, available resources, and the relevance of existing standards. It seems recommendable to start with existing modelling languages and tools, and, in parallel to reference model development identify their deficits, and, based thereupon, develop improved modelling languages and tools.

Furthermore, the initialisation of first open model projects requires a course of action aligned to the considerations above. In [FrSK07b], we discuss options for later phases in the life-cycle of open model projects; here we focus on initialising a first project. To reach a critical mass of participants, it is essential to convince firms, their employees as well as researchers. Following FOSS, useable and useful artefacts, i.e., reference models, their documentations, use cases etc., are among the most convincing means. Convincing initial successes are known to function as catalyst for developer and user participation [KoSF06]. Considering current entry barriers (see Sec. 1), the development of initial artefacts can hardly be left to individual researchers or firms. Rather, combined efforts of a few research institutions and selected practitioners are required. Only after their collaboration has produced usable and useful artefacts, a wider audience is likely to perceive open model projects as viable. The latter implies advertising open model projects through academic as well as professional channels. It also suggests obtaining endorsement from industry associations and scholarly organisations.

Initialising first open model projects implies specifying and evolving role concepts and role-specific incentives [KoSF06]. Researchers are likely to take on the roles of model designer, model maintainer and model reviewer. Especially the role of a model designer requires the widely accepted recognition of researchers' contributions as equivalent to regular publications. As a first step it is necessary to establish specific review processes for contributions to open model projects similar to those for peer-reviewed journals. Firms and their employees are likely to take on the roles of model evaluators and (active) model users. We see domain experts primarily in the role of model evaluators and IT personnel in the role of model user. Their primary incentive originates from the benefits attributed to the use of open reference models. A secondary incentive is seen in the exchange with the scientific modelling community.

6 Conclusions

The present contribution investigates whether FOSS development serves as a suitable orientation for overcoming the reference model paradox. Our findings suggest that community-driven collaborative modelling projects based on the notion of open reference models – or simply, open models – resolve the paradox. However, the analysis draws analogies to FOSS, a phenomenon only partly understood at present, and, thus, our investigation has to remain speculative at times. We, therefore, state our – preliminary – findings as working hypotheses and aim at stimulating a debate on community-driven collaborative modelling as a new organisation of reference modelling research and practice.

At the same time, we do not want to conceal that our intention is to promote the idea of an open model initiative. Aside from open questions and associated risks, we hold the view that such an initiative should be established as it is promising for a number of reasons: It strengthens IS research at its core and supports IS teaching. It fosters collaboration between Information Systems, Software Engineering, Computer Science, and the business- and management-related sub disciplines such as organisation, finance, and operations. It can, furthermore, be seen as a new model for organising research following the ideas of open science [Guad05]; and as a catalyst for disseminating research results to business practice.

Our assessment is backed by the feedback from academics and practitioners alike and by the endorsement from the Special Interest Group on Modelling Business Information Systems (SIG MoBIS) and from the Joint Interest Group on Modelling (JIG Modelling) at the German Informatics Society. In fact, after the idea of open models was first published in June 2006 [KoSF06], we launched openmodels.org as a hub for further discussions in November 2006.

References

[Aspe05] Aspeli, M.: Plone: A model of a mature open source project. M.Sc. Thesis, London School of Economics. London, U.K. 2005.

[BeSc04] Becker, J.; Schütte, R.: Handelsinformationssysteme: domäneorientierte Einführung in die Wirtschaftsinformatik. Redline Wirtschaft bei Verl. Moderne Industrie, Frankfurt am Main 2004.

[Benk02] Benkler, Y.: Coase's Penguin, or, Linux and The Nature of the Firm. In: The Yale Law Journal 112 (2002) 3, pp. 369-438.

[BoRo03] Bonaccorsi, A.; Rossi, C.: Why Open Source Software can succeed. In: Research Policy 32 (2003) 7, pp. 1243-1258.

[Broc04] vom Brocke, J.: Internetbasierte Referenzmodellierung – State-of-the-Art und Entwicklungsperspektiven. In: WIRTSCHAFTSINFORMATIK 46 (2004) 5, pp. 390-404.

[BrBu06] vom Brocke, J.; Buddendick, C.: Reusable Conceptual models – Requirements based on the design science research paradigm. In: First International Conference on Design Science Research in Information Systems and Technology, Claremont, CA, 2006.

[Coff06] Coffin, J.: Analysis of open source principles in diverse collaborative communities. In: First Monday 11 (2006) 6.

[CrHo04] Crowston, K.; Howison, J.: The social structure of free and open source development. In: First Monday 10 (2005) 2.

[CrHA06] Crowston, K.; Howison, J.; Annabi, H.: Information systems success in free and open source software development: Theory and measures. In: Software Process: Improvement and Practice (Special Issue on Free/ Open Source Software Processes) 11 (2006) 2, pp. 123-148.

[CuKL98] Curran, T. A.; Keller, G.; Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice Hall, Upper Saddle River, NJ 1998.

[DKRW05] Dreiling, A.; Klaus, H.; Rosemann, M.; Wyssusek, B.: Open Source Enterprise Systems: Towards a Viable Alternative. In: Proceedings of the 38th Annual Hawaii

International Conference on System Sciences (HICSS-38) (2005).

[FeLo03] Fettke, P.; Loos, P.: Classification of Reference Models: A Methodology and its Application. In: Information Systems and E-Business Management 1 (2003) 1, pp. 35-53.

[FeLo04] Fettke, P.; Loos, P.: Referenzmodellierungsforschung. In: WIRTSCHAFTSINFORMATIK 46 (2004) 5, pp. 331-340.

[FeLZ05] Fettke, P.; Loos, P.; Zwicker, J.: Business Process Reference Models: Survey and Classification In: Workshop on Business Process Reference Models (BPRM 2005). Satellite Workshop of the Third International Conference on Business Process Management (BPM), Nancy, France, 2005, pp. 1-15.

[Fitz06] Fitzgerald, B.: The Transformation of Open Source Software. In: MIS Quarterly 30 (2006) 3, pp. 587-598.

[Fran07] Frank, U.: Evaluation of Reference Models. In: Fettke, P., Loos, P. (Eds.): Reference Modeling for Business Systems Analysis. Idea Group, Hershey, PA 2007, pp. 118-140.

[FrSK07b] Frank, U.; Strecker, S.; Koch, S.: 'Open Model' – ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung). ICB-Research Report. Institute for Computer Science and Business Information Systems (ICB), Duisburg-Essen University. 2007. http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReport08.pdf, cited 2007-11-14.

[FrSK07a] Frank, U.; Strecker, S.; Koch, S.: Open Model – ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik. In: Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler, B. (Eds.): eOrganisation: Service-, Prozess-, Market-Engineering (8. Tagung Wirtschaftsinformatik). Universitätsverlag Karlsruhe, Karlsruhe 2007, pp. 217-234.

[Gall01] Gallivan, M. J.: Striking a Balance between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies. In: Information Systems Journal 11 (2001) 4, pp. 277-304.

[Ghos05] Ghosh, R. A.: Understanding Free Software Developers: Findings from the FLOSS Study. In: Proceedings of the Perspectives on Free and Open Source Software. Eds.: Lakhani, K. R. Cambridge 2005, pp. 23-45.

[Guad05] Guadamuz, A. L.: Open Science: Open Source Licences in Scientific Research. Working Paper. University of Edinburgh, School of Law, AHRC Centre for Studies in Intellectual Property and Technology Law. 2005. http://ssrn.com/abstract=764064, cited 2007-02-24.

[HaOu02] Hars, A.; Ou, S.: Working for Free? Motivations for Participating in Open-Source Projects. In: International Journal of Electronic Commerce 6 (2002) 3, pp. 25-39.

[HeSc03] Healey, K.; Schussman, A.: The Ecology of Open-Source Software Development. Working Paper. University of Arizona, Department of Sociology. 2003. http://www.kieranhealy.org/files/drafts/oss-activity.pdf, cited 2007-11-09.

[HeNH03] Hertel, G.; Niedner, S.; Hermann, S.: Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel. In: Research Policy 32 (2003) 7, pp. 1159-1177.

[KeTe98] Keller, G.; Teufel, T.: SAP R/3 Process-oriented Implementation: Iterative Process Prototyping. Addison Wesley Longman, Harlow et al. 1998.

[KoSF06] Koch, S.; Strecker, S.; Frank, U.: Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach. In: Open Source Systems (IFIP Working Group 2.13 Foundation on Open Source Software), Como, Italy, 2006, pp. 9-20.

[Kris02] Krishnamurthy, S.: Cave or Community? An Empirical Investigation of 100 Mature Open Source Projects. In: First Monday 7 (2002) 6.

[Lakh05] Lakhani, K. R.; Wolf, R. G.: Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. In: Proceedings of the Perspectives on Free and Open Source Software. Eds.: Lakhani, K. R. Cambridge, MA 2005, pp. 3-21.

[LeTi02] Lerner, J.; Tirole, J.: Some simple economics of open source. In: The Journal of Industrial Economics 50 (2002) 2, pp. 197-234.

[LeTi05] Lerner, J.; Tirole, J.: The Scope of Open Source Licensing. In: Journal of Law, Economics, & Organization 21 (2005) 1, pp. 20-56.

[MaMA00] Markus, L.; Manville, B.; Agres, C.: What makes a virtual organization work? In: Sloan Management Review 42 (2000) 1, pp. 13-26.

[Mcco99] McConnell, S.: Open-Source Methodology: Ready for Prime Time? In: IEEE Software 16 (1999) 4, pp. 6-8.

[MoFH02] Mockus, A.; Fielding, R. T.; Herbsleb, J. D.: Two Case Studies of Open Source Software Development: Apache and Mozilla. In: ACM Transactions on Software Engineering and Methodology 11 (2002) 3, pp. 309-346.

[MoSp00] Moon, J. Y.; Sproull, L.: Essence of Distributed Work: The Case of the Linux Kernel. In: First Monday 5 (2000) 11.

[Pere99] Perens, B.: The Open Software Definition. In: Di Bona, C., Ockman, S., Stone, M. (Eds.): Open Sources: Voices from the Open Source Revolution. O'Reilly, Sebastopol, CA 1999, pp. 171-188.

[Raym99] Raymond, E. S.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly and Associates, Sebastopol 1999.

[RaTr99] Raymond, E. S.; Trader, W. C.: Linux and Open-Source Success. In: IEEE Software 16 (1999) 1, pp. 85-89.

[Rose03] Rosemann, M.: Enterprise System Management with Reference Process Models. In: Shanks, G., Seddon, P. B., Willcocks, L. P. (Eds.): Second-Wave Enterprise Resource Planning Systems: Implementing for Effectiveness. Cambridge University Press, Cambridge, U.K. 2003, pp. 315-334.

[Ross04] Rossi, M. A.: Decoding the “Free/Open Source (F/OSS) Puzzle” – a Survey of Theoretical and Empirical Contributions. Working Paper. Dipartimento di Economia Politica, Università di Siena. 2004. http://opensource.mit.edu/papers/rossi.pdf, cited 2007-02-21.

[Sche94] Scheer, A.-W.: Business Process Engineering: Reference Models for Industrial Enterprises. Springer, Berlin, New York 1994.

[ScNü00] Scheer, A. W.; Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. In: Oberweis, A. (Ed.): Business Process Management: Models, Techniques, and Empirical Studies. Springer, Berlin et al. 2000, pp. 376-389.

[ScWi06] Schelp, J.; Winter, R.: Method engineering: Lessons learned from reference modeling. In: First International Conference on Design Science Research in Information Systems and Technology, Claremont, CA, 2006.

[ShSR02] Sharma, S.; Sugumaran, V.; Rajagopalan, B.: A Framework for Creating Hybrid-OSS Communities. In: Information Systems Journal 12 (2002) 1, pp. 7-25.

[Stal99] Stallman, R.: The GNU Operating System and the Free Software Movement. In: Proceedings of the Open Sources : Voices from the Open Source Revolution. Eds.: Di Bona, C., Ockman, S., Stone, M. Sebastopol, CA 1999, pp. 53-70.

[Stal02] Stallman, R. M.: Free Software, Free Society: Selected Essays of Richard M. Stallman. GNU Press, Boston 2002.

[StGo06] Stewart, K. J.; Gosain, S.: The Impact of Ideology on Effectiveness in Open Source Software Development Teams. In: MIS Quarterly 30 (2006) 2, pp. 291-314.

[Tuom05] Tuomi, I.: The Future of Open Source. In: Wynants, M., Cornelis, J. (Eds.): How Open is the Future? Economic, Social & Cultural Scenarios inspired by Free & Open-Source Software. VUB Brussels University Press, Brüssel 2005, pp. 429-459.

[Wils99] Wilson, G.: Is the Open-Source Community Setting a Bad Example? In: IEEE Software 16 (1999) 1, pp. 23-25.

[YNYK04] Ye, Y.; Nakakoji, K.; Yamamoto, Y.; Kishida, K.: The Co-Evolution of Systems and Communities in Free and Open Source Software Development. In: Koch, S. (Ed.): Free/Open Source Software Development. Idea Group Publishing, Hershey 2004, pp. 59-82.

Ulrich Frank, Stefan Strecker

Chair of Information Systems and Enterprise Modelling
Institute for Computer Science and
Business Information Systems (ICB)
University of Duisburg-Essen
Universitätsstr. 9
45141 Essen
Germany
{ulrich.frank|stefan.strecker}@uni-duisburg-essen.de