Lightning talk: A Simple Profiling Framework for Software User-Producer Reciprocity Review

Carole Goble

School of Computer Science
The University of Manchester
Manchester, UK
carole.goble@manchester.ac.uk

The Software Sustainability Institute
UK

***Abstract*— Mismatches between users and producers of software, or indeed producers and funders of software, lead to misery. We propose a simple software project “reciprocity” framework from the perspective of the producer, covering 4 areas and 12 characteristics. By plotting the relative degree of some or all characteristics even subjective or rule of thumb values give project profiles. Such profiles can be useful tools for comparing projects against their own expectations and desires, to review and compare project types and identify user-producer reciprocity misalignments.**

***Index Terms*—software, profiling, reciprocity.**

I. Introduction

Software is fundamental to research: 7 out of 10 researchers surveyed in the UK report their work would be impossible without software [1]. Increasingly funding bodies and publishers are pressing for producers to make their software more available for review and for reuse. Furthermore, software producers are frequently required to show their software’s adoption beyond its original development team in order to raise revenues to continue development for their own use. Such adoption requires more than just a link to a binary; it requires active sharing, documentation, support and commitment to a level of service that perhaps had not been fully appreciated by the producers and may not be welcomed. In a recent study 77% of respondents cited “time to document and clean up” and 52% cited “dealing with questions from users” as barriers to sharing code [2]. “Sustainability debt” is a real cost without a clear bearer under our current project-based, novelty-first research software funding regimes [3]. Often as software matures, and community interest rises, the core funding drops.

Software users are also being pressed to more accountably credit software [4] and show greater responsibility for supporting the software they depend on. Users can have expectations that software should be freely available and support for it readily accessible. This can run counter to the resources available to the software producers and to their own interests. Perhaps the software was just a proof of concept or an incidental means to support the work of its originators without any planned subsequent use by others. In the controversial article [5] users of other’s data were characterised as “data parasites”. Software users who do not contribute or cite the software but demand support and attention might be considered in such unflattering terms. Responsibility for software support and sustainability should be borne by all parties.

Even when software producers explicitly set out to nurture and grow an open source community (rule 7 in Prlić and Procter’s “Ten simple rules for the open development of scientific software” [6]), and software users show willingness to contribute, this is still a resource hungry and difficult exercise. Managing contributions in a “commons production” setting is hard, sometimes incurring cost-benefit mismatches, motivational conflicts and significant integration costs [7] as well as costs in time and effort to oversee the process.

II. A Simple Software Reciprocity Framework

Mismatches in intentions and actuality between users and producers of software, or indeed producers and funders of software, lead to misery. To gather a coarse grained idea of the producer-user profile of a project we propose a simple reciprocity framework based on ideas by Crowston [8].

There are many elaborate software maturity frameworks, some already used by the UK’s Software Sustainability Institute: for example, the Software Sustainability Maturity Model (SSMM), OSS Watch Openness Rating, NASA Reuse Readiness Rating, CMM, and QSOS (see [9] for a useful list). Most include reuse and capability metrics and software and process quality reviews. We draw on aspects of the openness rating and the “ripeness” levels of the SSMM [9] but from the perspective of the *intentions* of users and producers of the software from the *viewpoint of the producer*. Our simple idea is: highlight reciprocity and service expectation mismatches; review projects’ expectations and desires; and compare against types.

Table I summarizes the framework, which is intended to be rough. By plotting the relative degree of some or all characteristics, even subjective or rule of thumb values give useful visual project profiles. For example, Fig 1 is the classic profile for software never intended or destined for use outside the walls of its originating lab. Fig 2 is the profile of a software platform developed as part of a computational infrastructure programme intended all along for wide-scale adoption.

III. What next?

Our reciprocity framework focuses on the intentions and behaviour of producers and consumers of academic software to quickly and simply classify projects; a preliminary and complement to the application of maturity models such as SSMM. We need further work to define the levels of each characteristic using analytical and empirical investigation. The UK's Software Sustainability Institute (http://www.software.ac.uk) Research Software Group, have consulted on 55 software projects, with another 9 currently underway. We intend to retrospectively review our consultancy cohort to see if this simple framework reveals informative patterns that chime with our experiences and to hone our characteristics and their levels.

TABLE I. A SIMPLE RECIPROCITY REVIEW FRAMEWORK

	Software Producer		Software User		
Adoption Intentions	Incidental	Built for me, stuck it out there	Me	Just my lab	Adoptive Community
	Familial	Built for people just like me	Family Friends	People I know	
	Fundamental	Built for others, many not like me	Acquaintances Strangers Rivals	People I don't know	
Control Intentions	Autocratic	I want/need control	Selfish	Download, use, no credit	Contribution intentions
	Cooperative	Community effort, credit shared	Contribute	Champion, credit, donate	
	Incidental	Not my core business, don't care about credit	Collaborate Sponsor	Co-resource, co-develop	

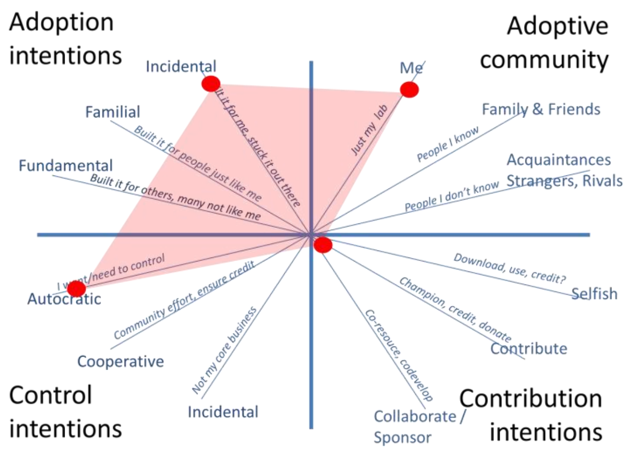

Fig. 1. Software only meant for its producers. Not a service.

During the Dagstuhl Perspectives Workshop 16252 "Engineering [of] Academic Software" held in June 2016 (http://www.dagstuhl.de/16252), we began to build on the ideas in this paper, maturity models and on Howison's organizational forms [10]. We are currently developing a set of dimensions to describe a set of software project types to use as a review tool for software producers, users and funders.

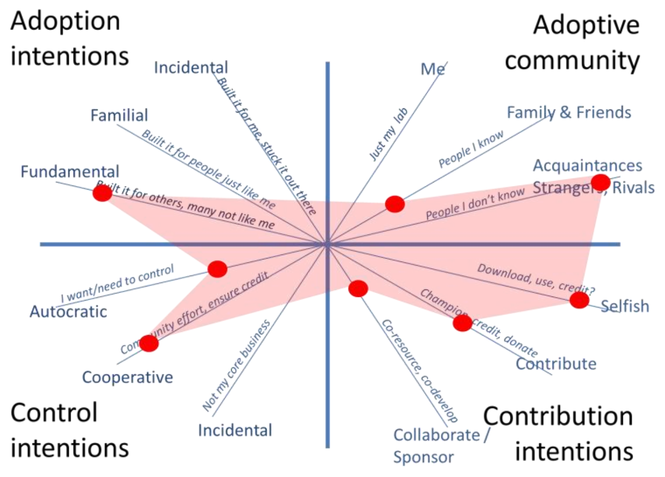

Fig. 2. Software intended for widespread use and as an outcome in its own right; a service and a possible candidate for further assessment.

ACKNOWLEDGMENT

This work was supported by EPSRC EP/H043160/1. We thank the WSSSPE reviewers for their insightful comments.

REFERENCES

[1] S. Hettrick et al (2014) UK Research Software Survey 2014, doi:10.5281/zenodo.14809

[2] V. Stodden (2010) The scientific method in practice: reproducibility in the computational sciences, MIT Sloan Research Paper No. 4773-10

[3] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S.M. Easterbrook, B. Penzenstadler, N. Seyff and C.V. Venters (2016) Requirements: the key to sustainability IEEE Software 33(1): 56-65. http://doi.ieeecomputersociety.org/10.1109/MS.2015.158

[4] A.M. Smith, D.S. Katz, K.E. Niemeyer, FORCE11 Software Citation Working Group. (2016) Software citation principles. PeerJ Preprints 4:e2169v3 https://doi.org/10.7287/peerj.preprints.2169v3

[5] D.L. Long and J.M. Drazen (2016) Data sharing, N Engl J Med 374:276-277 doi: 10.1056/NEJMe1516564

[6] A. Prlić and J.B. Procter (2012) Ten Simple Rules for the Open Development of Scientific Software. PLoS Comput Biol 8(12): e1002802. doi:10.1371/journal.pcbi.1002802

[7] J. Howison and J.D. Herbsleb (2013) Incentives and Integration In Scientific Software Production in CSCW '13 Proceedings of the 2013 conference on Computer supported cooperative work: 459-470, doi:10.1145/2441776.2441828

[8] K. Crowston, K. Wei, J. Howison, A. Wiggins (2012) Free/Libre open-source software development: What we know and what we do not know, ACM Computing Surveys 44(2), doi:10.1145/2089125.2089127

[9] R. Gardler (2010) Software Sustainability Maturity Model http://oss-watch.ac.uk/resources/ssmm (accessed 12 Aug 2016)

[10] J. Howison (2015) Organizational forms, http://www.slideshare.net/jameshowison/scisoftdays-talk-howison-spreading-the-work-in-software-ecosystems/18