International Journal of
Information Systems and Project Management
ISSN (print):2182-7796, ISSN (online):2182-7788, ISSN (cd-rom):2182-780X
Available online at www.sciencesphere.org/ijispm

Lifecycle management in government-driven open source projects – practical framework

Katja Henttonen
VTT Technical Research Centre of Finland
Tietotie 3, 02150 Espoo
Finland
www.shortbio.org/katja.henttonen@vtt.fi

Jukka Kääriäinen
VTT Technical Research Centre of Finland
Kaitoväylä 1, 90530 Oulu
Finland
www.shortbio.org/jukka.kaariainen@vtt.fi

Jani Kylmäaho
National Land Survey of Finland (NLS)
Opastinsilta 12C, 00521 Helsinki
Finland
www.shortbio.org/jani.kylmaaho@nls.fi

Abstract:
In many parts of the world, public sector organizations are increasingly interested in collaborating across organizational (and even national) boundaries to develop software solutions under an open licence. However, without sound lifecycle management practices, the full benefits of open collaboration are not achieved and projects fail to achieve sustained success. This paper introduces a lifecycle management model and framework for government-driven open-source projects and reports about its use in a real-life case study. Our focus is on lifecycle management activities which take place between deployment and end-of-life. The framework was developed iteratively through a series of focus group discussions with representatives of public sector organizations. After the framework had been taken into use in our real-life case project, individual qualitative interviews were conducted to collect experiences on its benefits and weaknesses. According to the initial evidence, the deployment of the framework seems to have brought concrete benefits to the project, e.g. by contributing positively to community growth, software quality and inter-organizational learning.

Keywords:
public information systems; open source; open-source software; free software; e-government; public sector; software lifecycle management; software evolution; information systems; public sector.

DOI: 10.12821/ijispm050302

Manuscript received: 1 May 2017
Manuscript accepted: 10 September 2017

1. Introduction

In many countries, governments agencies have started to open up bespoke software developed with public funding, often by releasing it under an open source license [1]–[3]. This may stem from governments' desire to spur innovation (by letting all citizens to gain from software at no additional costs) and/or to improve transparency (e.g. by making source code of an electronic voting system subject to public scrutiny) [2]. Another key rational behind open sourcing is a belief that other government agencies, who have similar software development needs, can reuse the software [1]–[3].

For example, in Finland, it was noticed that public sector organizations did not sufficiently co-operate on the field of bespoke software development [1]. In the absence of inter-agency collaboration, software vendors could charge each administrative unit a full price for the same or similar customizations and, thus, in the worst case, the same piece of code was purchased multiple times with tax-payer money [1]. For these reasons, the Finnish Ministry of Finance [4] and Public Administration Recommendations [5] have started to encourage public sector organizations to co-purchase bespoke software and publish it under an open-source license.

However, avoiding duplicate effort by open sourcing is not straightforward. It may be difficult for other organizations to exploit the source code purchased by one organization, e.g. due to lack of support and maintenance, multiple parallel development paths and uncertainty on the future development direction [1], [3], [6]. Therefore, there is a need to build public sector communities around these software initiatives to collaboratively manage the lifecycle [1], [2], [6], [7].

To address these issues, Kääriäinen et al. [1] developed a model where public sector agencies co-produce and co-maintain open-source software products together. However, at the time, the model had not been tested in any organization and its presentation remained abstract. This article concretizes the model introduced by Kääriäinen et al. [1] and demonstrates its practical value. The aims of the study are two-fold: firstly, to develop a practical framework that facilitates adoption of the model and, secondly, to use the framework for organizing collaborative lifecycle management in a real-life case study. The case study is an open source spatial data visualization software called Oskari, which is currently being co-produced by more than ten public sector organizations and companies in Finland.

The authors have studied the concept of the lifecycle management previously focusing on the development phase of the software (SW) product [8]. The emphasis of this article is on the lifecycle management actions taken after the implementation of the first software version i.e. how the developed SW product under the operation and maintenance could be collaboratively maintained and further developed by the group of public sector organizations.

The article is structured as follows. The next section covers theoretical background and related work, reviewing different approaches to change/lifecycle management in software production and summarizing studies on open-source lifecycle management and government-driven open-source software development. The third section introduces the model on which the framework has been built. The fourth section introduces the research approach and methodology. The fifth section introduces the practical framework which supports the deployment of the model. The sixth section demonstrates the deployment of the model and the framework of the Oskari project and reports on the experience gained. Finally, discussion and conclusions are drawn.

2. Theoretical background and related work

2.1 Lifecycle management and software evolution

Software lifecycle management (SLM) is herein understood as a process of coordinating activities and managing resources (e.g. people, money, documentation, technical artefacts) during the entire lifecycle of a software product, from initial ideation to retirement [9]. This definition comes from the application lifecycle management (ALM) literature, but similar issues have also been addressed by studies on software configuration management (SCM) and software evolution. However, SLM is different from software product management (SPM), which focuses solely on managerial actions taken before customer delivery of a software product [10].

Application lifecycle management (ALM) is a relatively new concept [11]. Chappell [12] presents ALM as a combination of three functions: governance, development and operations – and three milestones: (start of) ideation, deployment and end-of life. Development takes place at the beginning of the lifecycle, between ideation and deployment, and then periodically (after deployment) when the application is updated. Operations, which involve monitoring and deployment of updates, always happen after deployment of the first software version. Governance, which means supervising the software's evolution towards predefined goals, is needed during entire lifecycle. The emphasis of this article is on lifecycle management actions which take place between deployment and end-of life. Out of the three functions, most attention is given to governance but development and operations issues are also touched.

Software Configuration Management (SCM) is a much older discipline and can be seen as the basis upon which ALM is founded [13]. SCM is essentially about controlling and tracking changes to the software, and it has been discussed in the literature for more than three decades [14]–[17]. SCM research has significantly impacted software engineering practices [18]. The (sub)areas of SCM provide techniques for change control boards, defect tracking, build and release management, versioning and team/workflow management, for example [15], [17].

The term software evolution was originally used to differentiate from software maintenance which, at the time, was seen as a post-deployment activity consisting only of bug fixes and minor adjustments [19]. Early software evolution literature [20] noted that requirements continue to change and software needs to be adapted during its entire lifetime. Because the idea of iterative software development has become widely accepted, some authors use the terms software maintenance and software evolution synonymously [19]. However, there are two prevalent perspectives to software evolution, dubbed 'what/why' and 'how' by [19]. The former (what/why) refers to academic research on the nature of the software evolution phenomenon, its driving forces and impact [19], [21]. The latter (how) refers to engineering studies on practical means (e.g. technology, methods, tools) to direct, implement and control software evolution [22]. The focus of this article resembles the 'how perspective' on software evolution. However, the authors felt that when talking about the purposeful actions taken to ensure that a software product develops in the desired direction, lifecycle management is a more suitable term.

2.2 Open-source software production in the public sector

The term open source can be used to refer either to a licensing model or a software-development model [23]. Open-source licensing allows anyone to access the source code of the software, modify the software as desired and share it with others by redistributing a modified or unmodified version [24]. As a development model, open source refers to projects where relatively loosely coupled individuals and organizations collaborate to co-develop a piece of software together, typically working over the Internet in a distributed environment [25], [26]. Practices typically associated with open-source development include agile development, meritocratic governance and volunteer participation [26] for example.

During the last decade, government agencies all over the world have also become interested in open-source software development. Several communities or repositories for public sector open-source software development have sprung up, e.g. European-level Joinup, Finnish Yhteentoimivuus and Government GitHub. Joinup is meant for sharing and reusing open-source software, semantic assets and other inter-operability solutions for public administrations. Yhteentoimivuus is a delivery channel for public sector interoperability assets administrated by the Finnish Ministry of Finance. Government GitHub allows government agencies to share code and data on the social coding platform GitHub. While some government-driven open-source development projects have been abandoned, many others are active and continue to grow: CONNECT Health, OskarEMR, WorldWind and CAMAC, for example. Surprisingly, while there is a large body of research on open-source adoption by government organizations, e.g. [27]–[30], very few studies have looked at open-source production by government organizations. The latter are reviewed below.

Mergel [3] studied a context where government agencies share code through a common repository but do not form an open-source project or otherwise co-ordinate collaboration. The most common activity was found to be forking: participants copied the code release of another organization and then possibly modified it for their own needs internally [3]. Contribution back to the original project was not usual [3]. In other words, participants seemed to favor the

relatively passive process of 'copy and reuse' over active collaboration. These findings on government code-sharing mechanisms are in line with [1]: the absence of lifecycle management practices leads to multiple forks and therefore the potential benefits of collaborative development are not fully achieved.

Bryant and Ramsamy [31] analyzed ten open-source projects where public sector agents are key contributors. There are also few academic case studies on specific open-source collaborations in the public sector [32]–[34]. Studies demonstrate that organizational and political factors play a large role in government open-source projects [31], just like many other IS projects in public sector [35]. Success factors include trust between key stakeholders, skilled in-house ICT personnel and steady financial support [31], [34]. Projects were found to be particularly vulnerable to sudden changes in political leadership and loss of key personnel [31], [34] . Some studies underline the importance of retaining the agility/flexibility inherent in the open-source model [31] while others emphasize managerial control [32]. Interest conflicts are also a common challenge. Feldman and Horan [32] note that public and private sector participants had varying perceptions of value propositions. Bryant and Ramsamy [31] report that end users experienced difficulties in making their 'voices heard' over bureaucrats whose budgets paid for the development.

2.3 The community-based software lifecycle management model

Kääriäinen et al. [1] introduced the community-based software lifecycle management model (CO-SLM) aimed particularly at public sector organizations that finance and develop software collaboratively. In this model the term lifecycle management refers to actions taken after the implementation of the first software version. The model is applicable to free/open-source software development but also to other collaborative development models, as long as the licensing is sufficiently permissive to prevent vendor-locking and allows sharing of source code with other organizations. The model is depicted in Figure 1. The community has a common repository where the baseline version of the software product is stored. Each organization can use their own software supplier to take care of deployment, maintenance and customization of the software. However, they are encouraged to inform the rest of the community on changes made and contribute them back to the baseline version for integration. The integration work is coordinated by a 'product manager' and financed as agreed by the community (e.g. costs are equally shared by the community members). Parallel baseline versions are not maintained. The inclusiveness and openness of the development process are safe-guarded via a 'community manager' role.

According to Kääriäinen et al. [1], the core community consists of public sector organizations which have primary authority over lifecycle management decisions. Thus, the community becomes a key decision-making arena: individual government organizations can influence the development goals and evolution of the baseline software product by participating in the community. Very much like in 'traditional' open-source projects, the community can also become an arena for collaborative learning and knowledge sharing (e.g. sharing solutions to common deployment problems) and even collective innovation (e.g. ideating new functionality). Outside of the core community but still functioning as key partners are software companies that are tendered to develop the software [1]. However, in some cases companies may also participate in the community as full community members if the intention is to support the application of the software for the private sector as well (note that the model itself does not limit this). The community manages the software according to the lifecycle management plan initiated by the financier of the first version [1]. The plan defines who will do what and when in relation to the lifecycle management activities (e.g. documentation requirements, versioning model, change and release management practices and financing). Basically, this is a similar job that companies make for software products they own. Similarly, companies have product managers who are responsible for coordinating the lifecycle management actions to software products. However, when the group of public sector organizations start to jointly manage SW products the case is just more complex since there has to be found a consensus between the organizations what are the responsibilities, financing model, rights, etc.

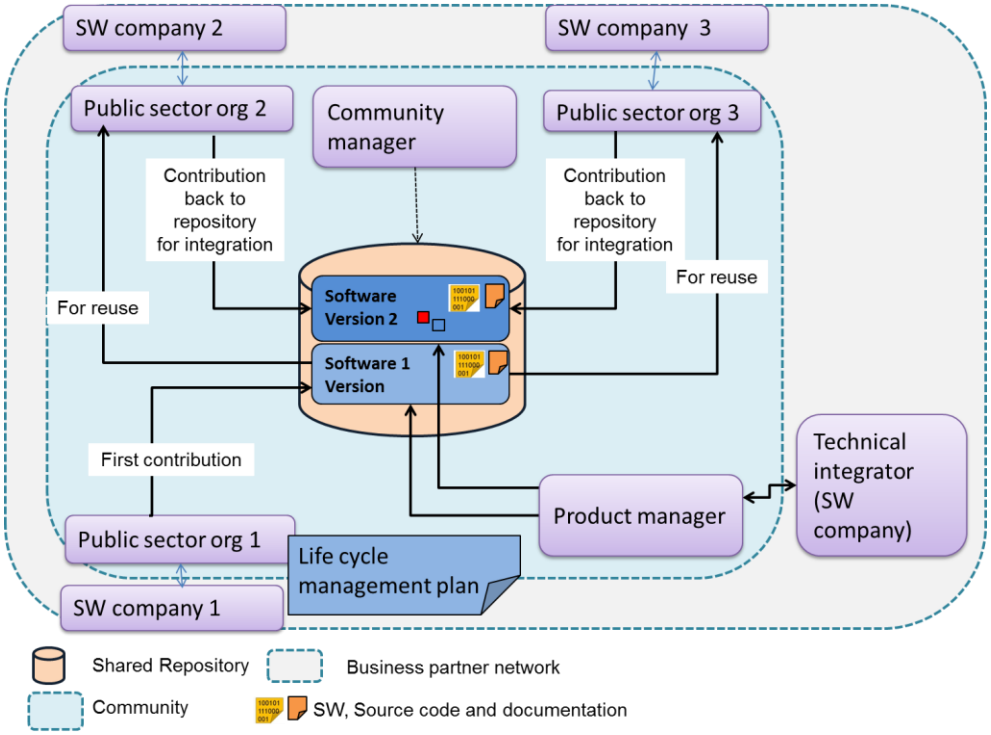

Fig. 1. Community-based Application Lifecycle Management Model

3. Research approach and methodology

The Ministry of Finance, our organization and a number of public sector organizations have collaborated in defining, piloting and deploying lifecycle management as depicted in Figure 2. The model creation process and the introduction to the models were published in [1]. This effort has since continued by piloting the planning of software lifecycle management in practice in the public sector. After successful piloting, the deployment of this model started in public sector organizations (the Finnish Ministry of Finance has accepted the model for production).

Prior and during the pilot phase, we developed the CO-SLM framework that is introduced in this article. The CO-SLM framework is a check list and documentation template to facilitate the definition of project-specific lifecycle management plans for software products. It helps software product communities to define a lifecycle management plan that describes who will do what and when related to the lifecycle management activities in the public sector software community environment. The framework has been tested and refined through deployment in real organizations.

The research presented in this article has an interpretive and an interventionary stance and, therefore, the approach could be described as an 'action case'. The term 'action case' was originally coined by Vidgen and Braa [36] to describe in-context information systems (IS) research which both aims to accumulate rich understanding on an organizational dilemma (interpretation) and to change the status quo in that organization (intervention). The interventionist phase typically takes place later in the research and involves the testing of the previously developed methods [36].

The CO-SLM framework was developed through iterative rounds of data collection and analysis. The primary data collection method was a series of six focus group discussions with information systems experts working in public sector

organizations. There were participants from both municipal (e.g. City of Espoo, the Association of Finnish Local and Regional Authorities) and national (e.g. State Treasury, Finnish Ministry of the Environment) levels of government. In addition, the representative from the Finnish Centre for Open Systems and Solutions (COSS) and a lawyer appointed by the Finnish Ministry of Finance participated some of the sessions. In the discussion sessions, participants were prompted to assess draft versions of the framework and their feedback was used to improve the framework iteratively. The point of saturation was reached after six consequent meetings. In addition to the focus group discussions, six Finnish, SME-sized software companies were asked to reply interview questions by email and clarifications were asked on the phone where necessary. These companies were selected for email interview due to prior collaborations with the public sector and consequent good knowledge of the domain. Complementary data collection methods also included few workshops with the Finnish Ministry of Finance and informal discussions with different stakeholders.

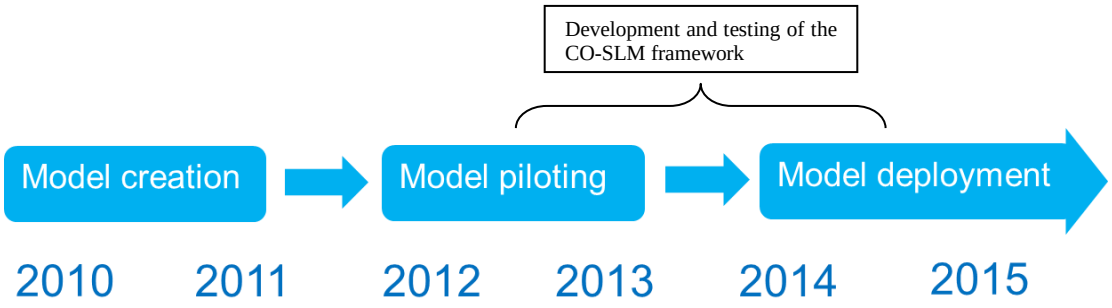

Fig. 2. Timeline for the development of the CO-SLM model

The testing and modification of the CO-SLM framework took place 'in situ' at multiple organizations during the years 2014 and 2015. For example, the National Land Survey of Finland (NLS) and the Ministry of Finance's JulkICT Lab project have adopted the model for their operation. This article explains and analyses its deployment within "Oskari", an NLS-led project which develops an open source geospatial toolkit. The reported experiences are based on two sources: 1) analytical observations from two of the authors who have been engaged in the Oskari project for a long time and (b) lengthy, semi-structured interviews of representatives from organizations who are key contributors to Oskari: National Land Survey of Finland (NLS), Finnish Transport Agency and The Finnish National Board of Antiquities. The interviewees were senior professionals who have co-ordinator responsibilities in the Oskari project, either in technical development or communications. Four out of five interviews were recorded and all were selectively transcribed. The thematic coding of the interview data followed a method called Template Analysis [37].

4. Framework for community-based lifecycle planning (CO-SLM framework)

In this chapter, we introduce a practical framework which helps with the application of the CO-SLM model into real-life situations where public sector organizations wish to develop software collaboratively. The framework focuses on the governance aspect of lifecycle management. The origins of the framework come from the SCM research area. One part of the SCM is a planning activity that forms an SCM plan [38]. The basic idea of the SCM plan is to define who is going to do what, when, where and how in relation to the configuration management [39]. When applied to the context of CO-SLM the goal is to help a consortium of public sector organizations to define what to manage, who will do the management, how the management will be done and how to finance the management and further development of a software product. Financing practices were included in the framework because collective purchasing and cost sharing is a significant and obvious concern for public sector organizations. Figure 3 depicts the four main elements of the product-management plan and Tables 1-4 present each of them in detail. The framework can be used as a check list and

template to form a lifecycle plan for a software product that needs governance during its lifecycle. The following four tables then describe each element in detail. Each element contains issues that need to be considered and documented for any software under management. Therefore, when applying the CO-SLM model and CO-SLM framework it should be borne in mind that each software product – and its associated community – is unique. Thus the model and framework must be applied to suit the context. This means making adjustments to terminology and content when applicable.

Fig. 3. Elements of the product-management plan

Table 1. What to manage?

Issues	Description
Name of the software	What is the name of the software program?
Licensing scheme	What are licensing terms for the source code and documentation?
User organizations	Which organizations will use the software?
Schedule for the first version	When is the baseline version of the software schedule to be ready?
Distribution channel	Where are the source code and documentation distributed?
Social media	Which social media channels are used by the project?

Table 2. Who will manage?

Issues	Description
Owner of the software product	Who 'takes care of' the software product? Who owns the copyright to the software?
Community structure and membership	How is the consortium of organizations structured? Are there community and steering groups? Who has the highest decision-making authority concerning the software product and its evolution?
Product manager (Development Co-ordinator)	Who supervises the software's evolution towards the commonly agreed goals? How is the role mandated? Coordinates the software product-related lifecycle management activities so that the software product evolves in the direction that serves the needs of the community and business.
Community manager (Openness Co-ordinator)	Who consolidates conflicts and protects the inclusiveness and openness of the development process? How is this role mandated? Coordinates the operation in the community. Checks that the licence is used as agreed by the community.

Table 2. Who will manage? (cont.)

Issues	Description
Repository maintainer(s)	Who maintains the shared repository containing the source code and documentation?
Technical integrator (Baseline Developer)	Who develops and maintains the baseline version of the software? Who is responsible for integrating the desired customisations to the baseline as agreed by the community? If the integrator function is outsourced, who does the tendering?
Providers of customisation and deployment services	Who can provide customisation and deployment services on the software product to individual member organizations?

Table 3. How to manage?

Issues	Description
Decision-making bodies	Responsibilities for making managerial and technical decisions regarding software development. How are the decision-making bodies (e.g. managerial board, change control board, steering group) organised, elected and assembled for a meeting?
Collaborative development approach	What are the key principles guiding collaborative development? How are the development efforts co-ordinated?
Road mapping	Who is responsible for creating and updating the roadmap documents? Who is responsible for accepting a new roadmap? Where are the documents located?
Change management	Who can initiate change requests, and how? Who analyses the change requests? How are requests prioritised? Who makes the final decision on what is included in the next software version?
Release management and versioning	How often are releases made? Who accepts a new baseline version for deployment? How are versions named/numbered?
Urgent bug fixes	Who/how to handle urgent bug fixes required to the baseline version already in deployment?
Communications	Who defines and supervises the community's communication strategy? What are the primary channels for internal and external communication?
Documentation	What documents are required and where are they located?

Table 4. How to finance?

Issues	Description
Co-ordination work	How are the efforts of the product and community manager financed?
Repository maintenance	How is repository maintenance financed?
Baseline development (Technical integrator)	How is the further development of the software product financed, including integration of external contributions? How to finance the bug fixes?
Deployment and customisation	Who pays for deployment and customisation work within an individual organization?
Community and steering group meetings	Who pays for organising and participating in the community meetings and steering group meetings?
New entrants	Who can join the community and how? Are there any joining fees?

5. Case study: Oskari software

This chapter presents a real-life case study where the CO-SLM framework has been applied. This chapter is structured as follows. The first sub-section introduces the Oskari case study. The second sub-section presents the lifecycle management plan for Oskari, which is based on the CO-SLM framework and briefly explains how the plan was made. The third sub-section reports on the benefits and challenges of lifecycle management as well as the experiences of using the framework in practice.

5.1 Introduction to the OSKARI case

Oskari is an open source software originally developed by the National Land Survey of Finland. Initially, Oskari was developed to offer easy-to-use browser-based tools to access and re-use information from various data sources, including the INSPIRE Spatial Data Infrastructure (SDI) and the Finnish National SDI. Oskari software has been adopted by about a dozen public sector organizations in Finland, including the City of Tampere, Finnish e-Government portal, Finnish Transport Agency and the Helsinki Regional Environmental Authority. Two major international co-operation projects utilising Oskari are currently running: European Location Services (ELS) and the Arctic Spatial Data Infrastructure (ASDI). The first independent Oskari installations are also emerging outside Finland: the National Land Survey of Iceland has set up Oskari, followed by Agency for Land Relations and Cadastre of the Republic of Moldovia.

Oskari makes it possible to view, visualize, analyze and even edit spatial data using just a web browser and standards-compliant APIs, such as OGC WMS (Web Map Service), OGC WFS (Web Feature Services) and OGC WPS (Web Processing Service). One of the most used features of Oskari implementations is the embedded maps functionality. It enables the user to choose applicable map layers and to create a map client using a WYSIWYG user interface without programming skills. The embedded map client can then be placed on any website in a similar manner as in Google Maps, just by placing a piece of HTML code into the website. The difference is that Oskari leverages standards-compliant APIs, which means that there are thousands of spatial data resources to choose from.

The Oskari network is a consortium of organizations that have entered into a formal agreement to co-develop the Oskari software. Oskari is published under open source licenses (MIT and EUPL) and therefore anyone can download the source code and utilise the software without joining the Oskari network. This means that anyone can try the software without committing to it or even without letting the network know about it, or use and extend it as they see fit. However, it is the appointed representatives of the steering committee member organizations who oversee which developed features or changes are integrated to the Oskari repository. The most important benefit of the steering committee membership is the ability to get support from other organizations and agree on the development goals together.

5.2 Lifecycle management plan for Oskari

The CO-SLM framework was used as a template and instructive guide when writing the lifecycle management plan for the Oskari software. The first draft of the plan was created by collecting existing practices found from websites and documents. Then the plan was discussed and refined to fill in any missing information. The plan template was also modified to be in the line with the terminology of software products and the software community. Finally, the plan was reviewed by the key members of the Oskari software team (the coordinator and the chairperson of the steering committee) and the plan was discussed and agreed (Version 1.0) in a steering committee meeting. The resulting plan is presented in the following tables 5-8 below.

Table 5. Basic information about the Oskari software

Issues	Details
Name of the software	Oskari
Licensing scheme	Open Source. Source code can be utilised using an MIT licence or EUPL licence.
User organizations	Public sector organizations, companies, non-profit organizations.
Schedule for the first baseline version	First public version was released 2011 (first version was financed by the National Land Survey Development Centre).
Distribution channel, repositories	Documentation, examples, etc.: http://www.oskari.org Source code: https://github.com/oskariorg General introduction to the software and the Oskari network (in Finnish): http://verkosto.oskari.org
Social media	Twitter: the @oskari_org Twitter channel reports new releases, bug and security fixes as well as events related to Oskari. The release plan and roadmap are presented on a Trello board (in Finnish): http://oskari.org/trello Slack: Slack is a team communication platform: https://oskari.slack.com

Table 6. Roles and organizations

Issues	Details
Owner of the software product	The Oskari network
Community structure and membership	The Oskari network is the development network for Oskari software that is open for anyone that signs the Memorandum of Understanding. Members (listed in Finnish): http://verkosto.oskari.org/oskari-verkosto/jasenet/
	Organization of the Oskari Steering committee: representatives of projects that exploit Oskari and sign the Integration agreement, coordinator (chosen by steering committee) and 1-2 representatives from the Oskari network member organizations (nominated annually by the Oskari network).
	Members (listed in Finnish): http://verkosto.oskari.org/oskari-verkosto/ohjausryhma/
Technical Coordinator (Product manager)	National Land Survey Development Centre (Jani Kylmäaho, Inkeri Lantta) http://verkosto.oskari.org/oskari-verkosto/koordinaattori
	The coordinator was selected by the Oskari steering committee. The coordinator coordinates (using the available resources) the software product-related lifecycle management activities so that the software product evolves optimally in the direction that serves the needs of the network and businesses. Furthermore, the coordinator facilitates the network and its activities, provides support to the projects utilising the software and works as a secretary for the steering committee. An architecture board meets 2 to 3 times per year to discuss and agree upon changes proposed to the technical architecture.
Community manager (Openness co-ordinator)	The National Land Survey Development Centre has the responsibility for this task as well.
Repository maintainer(s)	Technical coordinator
Integrator (Baseline developer)	The coordinator takes care of the integration work. The selected integrator is responsible for technical coordination, e.g. regarding the architecture of the software core. The integrator takes care of the integration work: coding, testing, version updates, documentation and any necessary IT support. The integrator reviews pull requests proposed by contributors, maintains repositories and core documentation and manages software versions, working in close cooperation with the coordinator.
Providers of customisation and deployment services	Each customer organization that applies Oskari software may select an IT provider for Oskari customisations without limitation. Customer organizations are encouraged to follow the architecture principles defined by the Oskari network if they wish to include modifications or extensions into the Oskari software.

Table 7. Practices for lifecycle management

Issues	Details
Tasks of the decision-making bodies	The Oskari network is open for anyone who signs the Memorandum of Understanding. The agreement describes the goals, tasks and decision-making practices for the Oskari network. The network communicates information about the Oskari software and its development, as well as discussing the future needs of the software. It has a mailing list and communicates actively in social media. Network members are invited to networking days (the steering committee schedules networking days at least once a year). Agenda for networking days: ▪ Status reporting and future activities ▪ Presentations of projects and activities around the software ▪ Selection of representatives of the steering committee Furthermore, developer meetings take place and the architecture group assembles 3 to 4 times each year. The goal of the developer meetings is to collect input that supports the development of the Oskari software core. The tasks of the Oskari steering committee are: ▪ Overseeing the network and planning activities ▪ Choosing the coordinator and setting the annual fees ▪ Prioritising the roadmap ▪ Communicating with the coordinator The steering committee also checks the status of the Oskari network (new members, etc.), communications activities, ongoing development projects, planned development projects, the roadmap and updated documents. The coordinator works as a secretary of the steering committee. The steering committee can invite the representatives of development projects to introduce and discuss their projects.
Collaborative development approach	The Oskari software is reused in development projects that need to create a web map application, a geoportal or to embed map clients into other web applications. The development project downloads the Oskari software and applies it; and further develops it, if needed. The development needs will be discussed with the coordinator and other development projects to avoid overlapping development work. The project is requested to follow the Oskari architectural principles and to provide modifications (Oskari open source licence) back to the Oskari network for integration. The Oskari steering committee decides what will be integrated into the next public Oskari release (or road mapped into future releases) based on the coordinator's proposal. Development projects are requested to document new source code to facilitate reuse (a documentation guide can be found on the Oskari website). The coordinator is responsible for checking the documentation during integration.
Road mapping	The coordinator maintains the Oskari roadmap (short-term roadmap and longer-term (1 year) roadmap) and is responsible for introducing new releases in steering committee meetings. The steering committee has the responsibility of checking and agreeing on any major changes before release. The roadmaps can be found at: ▪ http://oskari.org/trello (in Finnish) ▪ http://www.oskari.org/documentation/development/roadmap (in English)
Change management	Requesting changes: Based on proposals from the development projects, the Oskari coordinator collects the new features that are proposed to be integrated into Oskari. Major changes in the software core are planned by the coordinator and presented to the Oskari architecture board, which discusses and agrees on the proposed changes. All other remarks and proposals will be reported as GitHub issues. Change proposal: The coordinator prepares the proposal. Change decision: The Oskari steering committee makes change decisions based on the coordinator's proposals. Change implementation: The coordinator arranges tendering for Oskari integration and core framework development work and makes acquisitions based on the tendering results. Tendering material templates are provided as guidance for other projects for their tendering purposes. The coordinator maintains the Oskari integration backlog in cooperation with the integrator. The coordinator updates the backlog based on the agreed integration tasks. The integrator is responsible for defining and scheduling more detailed tasks and setting foreseeable version numbers for backlogged items. The selected integrator takes care of the integration work: coding, testing, version updates, documentation and any necessary IT support.

Table 7. Practices for lifecycle management (cont.)

Issues	Details
Release management and versioning	After integration and testing, the integrator prepares a software version for release. The version numbering scheme is as follows: ▪ X = Major version with significant changes in architecture and/or APIs of the software: planned in the Oskari roadmap. ▪ Y = Minor version: planned in the Oskari roadmap. ▪ Z = Maintenance version: other small changes and bug fixes are marked with a maintenance version number. The coordinator introduces a new software release proposal (major or minor release) for the Oskari steering committee who check and agree on major integrations before the release. The steering committee is informed of any major plans to change the software core. The committee schedules the major changes to ensure smooth transition to the new version within member organizations. The coordinator takes care of any other necessary small changes and bug fixes (maintenance releases). Instructions on how to contribute to Oskari development using GitHub branches: http://www.oskari.org/documentation/development/how-to-contribute
Communications	The Oskari steering committee is responsible for the communication plan. The coordinator prepares change requests to the communication plan and the steering committee agrees them. The coordinator is responsible for implementing the communication activities as scheduled.
Documentation	Functional specification: http://www.oskari.org/ User guides: Developer guides for applying Oskari: http://www.oskari.org/ http://oskari.org/examples/rpc-api/rpc_example.html End-user guides: ELF service http://demo.locationframework.eu/ National Geoportal Map window: http://www.paikkatietoikkuna.fi/web/en/user-guide Installation and operational environment: http://oskari.org/documentation/ Technical description and instructions for Oskari developers: http://oskari.org/documentation/

Table 8. Financing practices

Issues	Details
Coordinator	Mostly integration fees collected from organizations who have signed the Integration agreement.
Community manager	Financed as part of the coordinator's work.
Integrator	Will be financed by the partners who have signed the Integration agreement (annual integration fee). The coordinator and development projects can also negotiate the sharing of integration costs if the integration fee turns out to be too low.
Oskari development	National Land Survey Development Centre/SDI team. Oskari network. Project funding.
Deployment and customisation	Each organization takes care of its own funding to apply the Oskari software.
Network and steering committee meetings	Each organization takes care of its own participation expenses. Meeting costs are covered by the integration fee.
New entrants	Oskari network: Free of charge. New members have to sign the Memorandum of Understanding. Oskari steering committee: steering committee members (development projects) sign the Integration agreement where they agree the annual integration fee. The steering committee agrees on the annual integration fee.

6. Experiences on deploying CO-SLM

This chapter reports experiences, particularly benefits, challenges and lessons from deploying CO-SLM model and framework within the Oskari project.

6.1 Product acceptance and quality

CO-SLM model and framework provided a strong governance model for cooperation by clearly defining responsibilities and processes. As one of the interviewees point out, "lifecycle management is what actually turns an open software application into a software product…". Being a product means the availability of technical support and documentation, version schemes and roadmaps for future development, for example. This allows potential users to evaluate the suitability of the software to their current and future needs. The productization also includes communications and marketing activities, which – together with the robust management model – have helped to improve the "brand" of the software and make it more attractive to adopt. Overall, many interviewees talked about the Oskari brand and its importance to project acceptance.

The Oskari project had recently entered incubation process to join the OSGeo foundation, a not-for-profit legal entity supporting the open source geospatial community. This is expected to further improve the Oskari brand and acceptance of the software product, also internationally, but the process is in early stage. Generally, presence on platforms like GitHub and OSGeo where curious outsiders can explore the software without making financial or other commitments, is seen as a key to identifying stakeholders and growing the user base. One of the interviewees expressed this as follows:

> OsGEO, GitHub and other platforms where anybody can participate in the discussion are really good. You do not have to identify all stakeholders in advance, but just throw out something and interested parties will come to you. We have received inquiries from as far as Moldova…[ ]... If you want "fresh blood" [into the project], it is great that people can start following you without commitment and then deepen their involvement gradually.

The CO-SLM model has also helped to improve non-functional qualities of the software, particularly adaptability and extensibility. When a software is developed by a single organization alone, hectic demands and limited resources can cause focusing on immediate user needs at the expense of long-terms software quality, e.g. architecture design that allows software to adopt to future needs. Consequently, the software becomes hard to maintain within a single organization and impossible to share with other organizations without significant refactoring. However, CO-SLM model forces the owner to look at the software from a wider perspective, beyond their own immediate use cases. Each modification to the baseline version is considered from the viewpoint of multiple organizations, leading to improved adaptability. One informant explained:

> Our understanding [of software design] has broadened so much after we started talking with other organizations who have similar needs. It was a bit like 'oh, right, we do not have to reinvent that wheel'. We have learned that we can develop things collaboratively even though the needs are not exactly the same".

The CO-SLM model has also helped to secure resources for developing project-wide testing methods and tools available to all member organizations. This has reportedly decreased the number of software bugs in new releases.

6.2 Resource pooling

6.2.1 Human resources

For more than two decades, public sector organizations in Finland have been inclined to outsource all their software-development activities. This has caused a shortage of skilled in-house ICT personnel in many organizations, making it harder for them to take responsibility for software development and lifecycle activities. For example, when

organizations buy Oskari implementations from software companies, they may not know how to communicate key architectural principles to the companies or conduct tendering process in a manner that obligates companies to follow them.

This has sometimes led to poor-quality code contributions, e.g. new functionalities have been placed in illogical parts of code structure and/or interfaces are used in a non-standardised way. The resulting problems require significant effort from the technical coordinator (NLS) who underlines that, in the future, more effort must be put into ensuring a common understanding of proper architecture principles and their inclusion in the request for tenders.

The alternative strategy to address human resource deficits has been to acquire manpower from software companies in a form of 'body shopping'. This differs significantly from a process where public sector organizations place a request for tenders (RFT) for software implementation. As the bidders aim to offer the lowest possible price, no requirements apart from those explicitly mentioned in the RFT will be taken into account. According to both the interviewees and prior studies [35], [40], the heavy-weight requirements specification (for RTF) makes it difficult to incorporate new ideas afterwards and can thus hinder innovativeness. Oskari community has noticed that, in complex development cases, it is often better to tender for individual developers instead of tendering for specific implementations. This approach has enabled agile software development processes and intensified knowledge exchange between public- and private-sector organizations. In this model, the leadership of the software-development process stays entirely with the public sector, which again requires specific skills, different from those required by mere software acquisition.

It was also repeatedly noted that, while CO-SLM model does indeed require new skills, it also creates an environment for inter-organizational learning and thereby helps building new skills. For example, interviewees gave statements like "inter-organizational learning is a key benefit [from Oskari participation]", "even organizations who do not contribute code have provided much valuable inputs [of skills and ideas]" and "we have learned so much just by talking to other organizations with similar needs".

6.2.2 Financial resources

Because Oskari is open source licenced, any organization could download it and use it without participating in development expenses. However, the majority of organizational users have chosen to pay an integration fee. The payment ensures them a membership in the steering group and an opportunity to influence the future development of the software. By participating in the decision-making, organizations can ensure that the software will continue to meet their needs in the future. This has been enough to motivate organizations to contribute financially, and, thus, open source licensing has not lead to a significant 'free riding' problem.

Despite this, financing the Oskari baseline software development has not been easy. The relatively low integration fee (currently EUR 5,000 per organization annually) has been sufficient to cover the integration, co-ordination and communication activities of the Oskari community but not maintenance of the baseline software. Steering committee members felt it was impossible to increase the fee without forcing member organizations to go through a significant amount of bureaucracy. For long time, the development of the baseline version was paid for entirely by the National Land Survey of Finland, which made the project extremely dependent on a single organization. However, the increasing number of participants has recently improved finances and Oskari community is moving towards a model where it is less dependent on NLS funding.

In general, interviewees felt that the CO-SLM model has enabled significant savings because development cost can be split with others. One of the informants put it as follows:

> One of the major goals of this collaborative development model has been to save funds. I feel that we have achieved that… We can go to a steering group and split up tasks [between organizations], like 'you do this and I do that'… If we did not have this collaboration, we would have had to pay everything alone.

In practice, there were two ways to finance major extensions to Oskari: (1) some member organizations pay for Oskari extensions alone but comply with architectural rules and share them with others for free and (2) some organizations

form ‘mini consortia’ with other organizations who needed the same functionality and make the requirement analysis, tendering and financing jointly. The latter was considered as more mature form of co-development but required more inter-organizational communication and trust. One interviewee explained:

> The first level is that each organization manages their own projects but follows some commonly agreed principles. Meanwhile, others are waiting to get their hands on it. This is the most common way because it is fast and easy if you have money [within one organization]. The second level involves collaborative financing; it is much more complex and requires trust. One organization is chosen as a leader and then the leader organization makes consortium agreement with other organizations to co-finance and co-develop something together.

6.3 Project sustainability

Because big money is circulated in public sector ICT procurement, successful new models, which create savings for governmental organizations, unavoidably shrink revenues of some companies. Consequently, Finnish Location Information Cluster, an advocacy group of some established companies offering geospatial solutions, has been very critical of the Oskari project and tried to create political pressure against it. While no significant harm has been caused, aggressive industrial lobbying was noted to be a risk factor which can negatively influence sustainability of any government-driven open source project. CO-SLM approach partially helped to tackle the issue, e.g. by resourcing communication and public relations (PR) activities.

However, with Oskari, the biggest sustainability challenge is to decrease the project’s dependence on the coordinator, NLS, and thus make it less vulnerable to changing management interests and/or shifts in key personnel within that organization. This was expressed in several interviews, for example as follows: “even though NLS has been a primus motor in the start, there is no particular reason why it should remain as a primary or principal actor” and “other actors must take more responsibility because we [the project] should not be overly dependent on NLS”.

Significant informing and marketing effort has been undertaken to attract more organizations to the Oskari network. When more organizations are participating, more financing will come in and relevant technical knowledge will be distributed among multiple organizations and people. If the software is strategically important to a sufficiently large number of organizations, the development will continue even if the NLS decides to drop out. The project is now entering a new phase as the co-ordinator role is planned to be shifted from NLS to an outsourced project organization whose costs are covered with integration fees.

7. Discussion

7.1 Lessons for researchers and practitioners

For practitioners in the public sector who consider engaging their organizations in collaborative open source projects, the case study highlights the importance of ensuring sufficient in-house IS skills. This is in line with prior literature pointing out in-house IS skills as a key success factor to open source and other agile projects on the public sector [32], [35]. Even though it is possible and often recommended [34] to exploit external experts, open source development requires the public sector to take on the responsibility of a software owner. This requires both sufficient technical competence and knowledge in software lifecycle management. To support the latter, the CO-SLM framework acts as a document template for planning software governance and lifecycle activities.

The second lesson for practitioners has to do with the importance of enabling community growth. A ‘critical mass’ of active organizational users helps to ensure steady funding and guarantee project continuity, even if one dominant organization drops out. This is also in line with prior studies which have emphasised versatile developer and donor bases as a success factor to all types of open source projects [41], [42]. In part, CO-SLM supports community growth by making the software more attractive to new users. This is because clarified governance processes and responsibilities make the whole process more predictable and manageable. Software must also be ‘generic’ enough so that it can be

adapted to the diverse use cases for heterogeneous organizations. As the software is developed further, one must keep a multi-organizational perspective in mind.

For researchers and consultants, a key lesson relates to appreciating the huge diversity of organizations and software projects. Due to the heterogeneity of environments, it has proven pointless to develop a detailed predefined set of lifecycle management practices for all public sector driven open source projects. We noticed that a high flexibility of the framework is more important, taking, e.g. the form of 'check lists' on lifecycle management issues to be taken into account. Each project can then take the framework as a basis and develop lifecycle management practices suitable to their particular circumstances. If one wishes to develop 'best practices' on lifecycle management, one must focus on a particular software domain and type of application, not public sector driven OSS projects in general.

7.2 Limitations of the study and further research

While diverse stakeholders were involved in the drafting and development of the CO-SLM framework (see Section 3 for details), we did not interview all members of the Oskari network after its deployment. Because we had an opportunity to interview only people from three heavily-engaged organizations (see chapter 3), the results are biased towards their perspectives. We acknowledge that 'peripheral' members of the Oskari community may have different perspectives that are not visible in this study. We also understand that a single case study is not enough to make definitive conclusions regarding the applicability of the framework in diverse public sector environments. Our next step is to deploy the CO-SLM model and framework in other public sector open source software development efforts and, thereby, to gain further experience on their applicability in different organizational settings. We also hope to collect and analyse more qualitative and quantitative data on the supposed benefits of the CO-SLM approach.

8. Conclusions

This paper introduced the CO-SLM model and flexible framework developed for helping public sector organizations to follow sound lifecycle management practices in open source development projects. The model and the framework were successfully deployed in a real-life setting, where a dozen public sector organizations were jointly developing spatial data analysis software under an open source licence. The adoption of CO-SLM benefited the software project by encouraging community growth, improving the 'image' of the software and enhancing software quality, especially regarding software maintainability and extensibility. Challenges stemmed from deficit software development and acquisition skills in some organizations and insufficient funding due to relatively low membership fees. Furthermore, the study shows that a project's financial and technical dependence on the leading organization should be decreased in the future to lower risks and ensure long-term sustainability.

9. References

[1] J. Kääriäinen, P. Pussinen, T. Matinmikko, and T. Oikarinen, "Lifecycle Management of Open-Source Software in the Public Sector A Model for Community-Based Application Evolution," ARPN Journal of Systems and Software, vol. 2, no. 11, pp. 279–288, 2012.

[2] J. C. Colannino, "Free and Open Source Software in Municipal Procurement: The Challenges and Benefits of Cooperation," *Fordham Urban Law J.*, vol. 39, p. 903, 2012.

[3] I. Mergel, "Open collaboration in the public sector: The case of social coding on GitHub," *Gov. Inf. Q.*, vol. 32, no. 4, pp. 464–472, Sep. 2015.

[4] Finnish Ministry of Finance, "Sade-ohjelma: open source approach," Helsinki, Finland: VM, 2012. Available: https://www.europeandataportal.eu/data/en/dataset/sade-ohjelma-avoimen-lahdekoodin-toimintamalli00

[5] JHS, *JHS 169 Use of Open Source software in Public Administration*. Helsinki, Finland: JUHTA (Advisory Comittee on Information Management in Public Administration), 2012.

[6] A. Nurmi, "Coordination of Multi-Organizational Information Systems Development Projects – Evidence From Two Cases," *J. Inf. Technol. Theory Appl.*, vol. 10, 2010.

[7] T. A. Pardo, A. M. Cresswell, F. Thompson, and J. Zhang, "Knowledge sharing in cross-boundary information system development in the public sector," *Inf. Technol. Manag.*, vol. 7, no. 4, pp. 293–313, Dec. 2006.

[8] J. Kääriäinen, "Towards an application lifecycle management framework," VTT Publications, no. 759. Espoo, Finland: VTT Technical Research Centre of Finland, 2011.

[9] S. Chanda and D. Foggon, "Application Lifecycle Management," in *Beginning ASP. NET 4.5 Databases*, Berkeley, USA: Apress, 2013, pp. 235–249.

[10] K. Vlaanderen, I. Van de Weerd, and S. Brinkkemper, "Improving software product management: a knowledge management approach," *Int. J. Bus. Inf. Syst.*, vol. 12, no. 1, p. 3, 2013.

[11] G. Weiß, G. Pomberger, W. Beer, G. Buchgeher, B. Dorninger, J. Pichler, H. Prähofer, R. Ramler, F. Stallinger, and R. Weinreich, "Software engineering - Processes and tools," *Hagenb. Res.*, no. 1, pp. 157–235, 2009.

[12] D. Chappell, "What is application lifecycle management," David Chappel and Associates, 2008, Available: http://davidchappell.com/writing/white_papers/What_is_ALM_v2.0--Chappell.pdf

[13] C. Schwaber, "The Expanding Purview Of Software Configuration Management", Forrester Research, 2009.

[14] E. H. Bersoff, "Elements of Software Configuration Management," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 1, pp. 79–87, 1984.

[15] J. Estublier, "Software configuration management," *Proc. Conf. Futur. Softw. Eng. - ICSE '00*, pp. 279–289, 2000.

[16] J. Koskela, "Software configuration management in agile methods," *VTT Publications*, no. 514. Espoo, Finland: VTT Technical Research Centre of Finland, pp. 3–54, 2003.

[17] M. E. Moreira, *Software configuration management implementation roadmap*. Chichester, England: John Wiley & Sons, 2004.

[18] J. Estublier, D. Leblang, A. Van Der Hoek, R. Conradi, G. Clemm, W. Tichy, and D. Wiborg-Weber, "Impact of software engineering research on the practice of software configuration management," *ACM Trans. Softw. Eng. Methodol.*, vol. 14, no. 4, pp. 383–430, 2005.

[19] T. Mens and S. Demeyer, *Software evolution*. Springer Berlin Heidelberg, 2008.

[20] B. W. Boehm, "A spiral model of software development and enhancement," *Computer (Long. Beach. Calif).*, vol. 21, no. 5, pp. 61–72, 1988.

[21] M. Lehman and J. C. Fernáandez-Ramil, "Software Evolution," *Softw. Evol. Feed. Theory Pract.*, vol. 27, no. 4, pp. 7–40, 2006.

[22] T. Mens, "Introduction and roadmap: History and challenges of software evolution," in *Software Evolution*, T. Mens and S. Demeyer, Eds. Berlin, Heidelberg, Germany: Springer, 2008.

[23] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/Libre open-source software development: What we know and what we do not know," *ACM Comput. Surv.*, vol. 44, no. 2, p. 7, 2012.

[24] Open Source Initiative, "The Open Source Definition," *Open Source Initiative*, 2013. [Online]. Available: http://opensource.org/osd.

[25] J. West and S. O'mahony, "The Role of Participation Architecture in Growing Sponsored Open Source Communities," *Ind. Innov.*, vol. 15, no. 2, pp. 145–168, Apr. 2008.

[26] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/Libre open-source software development," *ACM*

Comput. Surv., vol. 44, no. 2, pp. 1–35, 2012.

[27] B. Rossi, B. Russo, and G. Succi, "Adoption of free/libre open source software in public organizations: factors of impact," *Inf. Technol. People*, vol. 25, no. 2, pp. 156–187, Jun. 2012.

[28] M. Shaikh and T. Cornford, "Navigating Open Source Adoption in the Public Sector," *18th Am. Conf. Inf. Syst. (AMCIS 2012)*, pp. 1–10, 2012.

[29] J. Allen and D. Geller, "Open source deployment in local government: Rapid innovation as an occasion for revitalizing organizational IT," *Inf. Technol. People*, vol. 25, pp. 136–155, 2012.

[30] O. Jokonya, "Investigating open source software benefits in public sector," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2015, vol. 2015–March, pp. 2242–2251.

[31] D. Bryant and P. Ramsamy, "Public Administration Code Release Communities," Madrid, Spain: ONSFA, 2014.

[32] S. S. Feldman and T. A. Horan, "Collaboration in electronic medical evidence development: A case study of the Social Security Administration's MEGAHIT System," *Int. J. Med. Inform.*, vol. 80, no. 8, 2011.

[33] M. Liu, B. C. Wheeler, and J. L. Zhao, "On Assessment of Project Success in Community Source Development," *Proc. Twenty Ninth Int. Conf. Inf. Syst. (ICIS 2008)*, 2008.

[34] M. Liu, X. Wu, J. Leon Zhao, and L. Zhu, "Outsourcing of Community Source: Identifying Motivations and Benefits.," *J. Glob. Inf. Manag.*, vol. 18, no. 4, pp. 36–52, 2010.

[35] J. Nuottila, K. Aaltonen, and J. Kujala, "Challenges of adopting agile methods in a public organization," *International Journal of Information Systems and Project Management*, vol. 4, no. 3, pp. 65–85, 2016.

[36] R. Vidgen and K. Braa, "Balancing interpretation and intervention in information systems research: the action case approach," in *Information Systems and Qualitative Research*, 1997, pp. 524–541.

[37] N. King, "Template analysis. Qualitative methods and analysis in organizational research: A practical guide. ," in *Qualitative methods and analysis in organizational research: A practical guide*, 1998, pp. 118–134.

[38] A. Leon, *Software Configuration Management Handbook*. Boston, USA: Artech House, 2005.

[39] F. J. Buckley, *Implementing Configuration Managment, Hardware, Software and Firmware*. IEEE Standards Office, 1996.

[40] P. F. Manso and A. Nikas, "The application of post tender negotiation procedure: A public sector procurement perspective in UK," *International Journal of Information Systems and Project Management*, vol. 4, no. 2, pp. 23–39, 2016.

[41] I. Chengalur-Smith, A. Sidorova, and S. Daniel, "Sustainability of Free/Libre Open Source Projects: A Longitudinal Study," *J. Assoc. Inf. Syst.*, vol. 11, no. 11, 2010.

[42] M. Stuermer, G. Abu-Tayeh, and T. Myrach, "Digital sustainability: basic conditions for sustainable digital artifacts and their ecosystems," *Sustain. Sci.*, vol. 12, no. 2, pp. 247–262, 2017.

Biographical notes

Katja Henttonen
Ms. Katja Henttonen is working as a digitalization specialist at VTT Technical Research Centre of Finland. She has over 15 years of experience in software development gained in both the public and private sector. Since joining VTT, she has worked in various research projects around the following topics: open source systems, open innovation and collaborative economy. She holds a M.Sc. degree in ICTs and socio-economic development from the University of Manchester and is studying towards a Phd.

www.shortbio.org/katja.henttonen@vtt.fi

Jukka Kääriäinen
Dr. Jukka Kääriäinen works as a Senior Scientist at VTT Technical Research Centre of Finland Ltd in the Digital Transformation team. He has received PhD degree in 2011 in Information Processing Science from the University of Oulu. He has over 10 years of experience with product management, configuration management and lifecycle management. He has been involved in various European ITEA, ITEA2 and Artemis research projects.

www.shortbio.org/jukka.kaariainen@vtt.fi

Jani Kylmäaho
Mr. Jani Kylmäaho is currently employed at the National Land Survey of Finland, where his position is Head of Development for topographic data production. Jani worked as the product owner for the Oskari open source software until January 2017. He has been working with OGC services and both national and international SDIs for 15 years. He has extensive experience of open source software, agile methods, collaboration networks as well as INSPIRE implementation. Jani holds an MSc degree in Geography from the University of Helsinki, Finland.

www.shortbio.org/jani.kylmaaho@nls.fi