OPEN SOURCE DEVELOPMENT IN THE CANADIAN SPACE INDUSTRY

Author: Captain Jamie Wiecha (RCAF)
Contributors: Alex Caravaggio, Avry Harris, Jonathan Saunders
Defence R&D Canada Ottawa (DRDC), Ottawa, ON, Jamie.Wiecha@forces.gc.ca

ABSTRACT

This paper examines concepts for how the Canadian Armed Forces and the Canadian Space ecosystem can leverage *Open Innovation* to aid in future product development. This paper will focus specifically on a subset of open innovation called open source and collaborative software development. The aim is to provide a high-level overview from information abstracted from literature in hopes of educating the reader about potential consideration in future space projects. This research was conducted under the DRDC Space Common Operating Picture (S-COP) project to determine the viability of open innovation as a potential development strategy between The Department of National Defence (DND), commercial companies, academia and allied nations to enhance the Space Situational Awareness (SSA) capability in the Canadian Space Operations Centre (CANSpOC). While this paper utilizes the S-COP project as the application case study, the approaches are agnostic and can be applied to other space systems projects using a similar approach.

INTRODUCTION

"Software is the fabric that enables planning, weapons and logistics systems to function: it might be the only infinitely renewable military resource. In particular, DoD (US Department of Defense) must have a software environment that is easily adaptable to changing mission needs; this software must also evolve at lower cost and be delivered rapidly so it can be used when it is needed. This technological evolution entails a parallel evolution in acquisitions methodologies and corporate attitude to facilitate discovery, re-use, and modification of software across the DoD and U.S. Government. A new way is needed to develop, deploy and update software-intensive systems that will match the tempo and ever-changing mission demands of military operations" [1]

This assessment was conducted under the DRDC Space Common Operating Picture (S-COP) project to determine the viability of open innovation as a potential development strategy for DND, commercial companies, academia and allied nations. While the S-COP has several mission areas, this paper examines the Space Situational Awareness (SSA) capability in the Canadian Space Operations Centre (CANSpOC). This paper will focus specifically on a subset of open innovation called *Open Source Development (OSD)* and *Collaborative Development*. A paper written by Weiss [2] titled "Business of Open Source," outlines key patterns of open source development strategies and was used as the foundation paper for this research. An extensive literature review was conducted to expand upon Weiss' paper and the knowledge obtained is presented within the context of space systems applications for DND. This paper is organized into three streams; (1) Levels of Engagement (Use, Contribute, Champion, Collaborate); (2) Architecture and (3) Licensing strategy.

The S-COP project was used as a case-study to analyze the system's development requirements against these three streams. This was to determine if open source or collaborative development methodology meets the unique requirements of a Defence organization. This paper provides an introductory framework to help understand how open source and collaborative development could be conducted in Canada and some considerations applicable to private sector, government, and Defence organizations.

SPACE COMMON OPERATING PICTURE

The S-COP is an applied research and development activity under the SSA project at DRDC. It is intended to provide understanding of the orbital situation to Canadian Forces operators, nominally located at the CANSpOC, to understand the activities, changes and implications of space enabled activated on Canadian Armed Forces operations. There are eight main mission areas of the CANSpOC which are summarized in Table 1.

• **Position, Navigation and Timing (PNT)**	• **Missile Warning**
• **Intelligence Surveillance and Reconnaissance (ISR)**	• **Satellite Communications Support**
• **Support to Terrestrial Operations** Joint Space Support Team	• **Specific Space Effects** Allied space data access
• **Space Operations** Satellite Operations Pre-emption and Shutter control Combined Space Operations Interference Response	• **Space Domain Awareness** Space Catalog Maintenance Conjunction Assessment Launches and Reentries Electro-magnetic Interference

Table 1 CANSpOC Mission Areas

RATIONALE FOR OPEN SOURCE / COLLABORATIVE DEVELOPMENT

Two important factors spurring the process of open innovation is the rising cost of technology development and shorter product life cycles in many industries. Companies are finding it increasingly difficult to justify investments in innovation. Open source and collaborative development methods may address both effects as it leverages external Research and Development (R&D) resources to save time and money in the innovation process. [3]. Some of the key benefits of open source and collaborative development include:

i) increased agility/flexibility;
ii) faster delivery;
iii) increased innovation;
iv) shared cost and risk of developing a platform (shared assets);
v) that collectives help its members achieve system-level objectives that none can achieve on their own;
vi) faster adoption and emergence of standards; and
vii) preventing "reinventing the wheel" and allowing combination of technical advances into one joint implementation.

"While the rate of change in technology grows exponentially, (a Defence organization) needs to continue to develop new capabilities ever faster. Gone forever are the days of a single contractor developing a system from scratch, uniquely matched to the required application. The need for commonality across platforms, interoperability between networks and shared functionality across organizations drive reusing what exists rather than reinventing the same basic constructs" [1].

ENGAGEMENT MODELS

When considering an open source project, it is important to understand how stakeholders will engage in the R&D of the project in order to aid in the project strategic direction and decisions. Weiss [2] categorizes the different methods of interaction with an open source project into four patterns: *Use, Contribute, Champion, and Collaborate.* Each of these Levels of Engagement (LoE) has their own unique advantages and disadvantages that a company can assess to determine their best method of engagement in an open source project. Depending on the complexity of the open source project, a member may have multiple LoE. The LoE describes the project development community, which differs from the end user community that uses the product. In this case, the end user would be a CANSpOC operator who employs the software product and does not actively develop the application.

Use

In the *Use* LoE, a company leverages the OSD project or components to build their product or service without contributing back to the project. They typically have little to no involvement or interaction in the project governance or direction. This LoE allows companies to reduce developmental costs and speed up creation by reuse of existing open source components [2].

In the S-COP program, the *Use* LoE could be viewed as companies or Defence organizations that would develop complements (add-ons or applications), but not contribute back to the core platform. These types of users would most likely be engaged by the project through a Software Development Kit (SDK) which would include relevant resources and information (e.g. Application Programming Interface (API), Graphic User Interface (GUI) resources, etc).

Contribute

Weiss [3] defines the *Contribute* LoE as when a company moves beyond the *Use* LoE and contributes back to the OSD project. "By taking a more active role in the projects they use by contributing to their communities, companies can build goodwill with these communities… companies generally achieve three goals by contributing back:

i) build trust with the community;
ii) influence the development of the project; and
iii) demonstrate their depth of competence" [2].

"Companies have also begun to recognize that making money from open source software while not giving anything back to the community or project is likely to ultimately result in the failure of their open source related products" [4]. Weiss [2] provides examples of potential contribution methods that include:

i) contributing new code;
ii) complementary activities including fixing bugs, customizing the software, & creating distributions;
iii) donation of existing code;
iv) non-code contributions that include financial, hosting, hardware support, project management, people, and funding; and
v) knowledge.

"Through their level of contribution, a member can ensure the core assets fit with their business goals. Members who contribute the most to a specific asset can expect to benefit when reusing the asset. A study of open source development found that contributors obtain private benefits from the development of shared assets that are not available to 'free riders' who only use the assets. These include learning, sense of ownership and control, and feedback from others on the contributed code. Contributors are also in a better position to

tailor their code to their individual needs, because the code that they contributed for general use may not be a good fit with someone else's needs" [5].

Champion

The *Champion* LoE is when "a company leads an open source project, and aims to create an ecosystem around it" [2]. Champions can be considered the owners of a project. Depending on the project's governance structure, the champion LoE has the most influence over the project's direction (both architectural and functional), how the project is licensed/released, and the stakeholders within the project. A champion could be a company, an organization, or an individual. According to Weiss, one of the primary roles of a champion is to develop and build a healthy community around the project, but during the initial start-up phase of the project, the champion has to support the community without the expectation of immediate returns [2].

In order to build a healthy community, Weiss explains that the champion has to create a credible promise to mobilize developers to join the project [2]. "Trust can be increased by developing key functionality early in a project to demonstrate that the project is doable and has merit" [5]. Dixon further reinforces this statement of credible promise by describing the importance of establishing legitimacy; "a company (project) builds up legitimacy with the community by:

i) giving to the community (code, documentation, participation in the discussion forum);
ii) clear licensing practice;
iii) clear process for making contributions;
iv) making decisions in the open, and
v) not treating community members as prospects" [6].

Establishing clear project governance documentation allows for greater trust and enables companies to confidently build their business models on top of the project and reduces the risk of uncertainty. "Successful leaders (champions) make a strong contribution and hold a central position in the community. Projects run by leaders (champions) who have demonstrated their technical skills and who have a record of past successes are generally more likely to succeed" [5]. Scott et al. reinforces the role of building a healthy community by stating "OTD (Open Technology Development) methodologies rely on the ability of a software community of interest to access software code or application interfaces across the enterprise. This access to source code, design documents and to other developers and end-users enables decentralized development of capabilities that leverage existing software assets. OTD methodologies have been used for open source development, open standards architectures, and the most recent generation of web-based collaborative technologies. The most successful implementations come from direct interaction with the end-user community. The open source software development model is successful because communities of interest involve both developers and users" [1].

The champion also has to determine and understand how much control it wants to have over the project based on the project's objectives. There is a fine balance between strongly maintaining the scope of the project while giving the community space to innovate and participate. By keeping tight control, the champion tries to advance their own goals, compared to looser control, which encourages growth where external participation and adoption from users/developers [7]. Weiss warns that the more control a champion has over a project, the more cost is incurred by the champion (e.g costs for development, hosting, documentation, and marketing) [7]. This decision of how much control the champion has will affect the licensing strategy, ownership of the code/project, and ability to maintain a code base that is not forked into several independent projects, thereby splitting the community and its resources.

Collaborate and the role of Collectives/Foundations

Weiss defines this last LoE as when "a group of companies collaborate to develop shared (i.e. nonstrategic) assets that each of them can use in its own projects" [2]. "It is often observed that somewhere between 50% and 90% of development effort is spent on creating software that does not differentiate a company from its competitors. Only the remainder differentiates a company from its competitors. This observation has motivated companies to acquire the non-differentiating parts of their software stack elsewhere, for example, as Commercial off-the-Shelf (COTS) or open source software. When such software is not available, or when a higher degree of control over the software is desired to enable more effective customization, organizations have joined efforts to create their own common software stack in a collaborative effort, making the result available to each other, or even to anyone else who wishes to use it" [5]. "For example, as a collective, a group of start-ups can deliver a complete solution to a customer, whereas individually they are only able to deliver pieces of the solution, which the customer has to integrate. Joining forces makes the group of start-ups much more competitive against large system integrators. Collectives can also collaborate to address common needs, allowing their members to focus on the differentiating features of their products. The more members a collective has, the more its members are able to share the load of meeting common needs" [5]. "Establishing collaboration isn't the same as creating a one-way communications strategy. Collaboration involves an easy interchange of ideas among many perspectives (including industry, academia and other government agencies offices and labs) to produce a better result than any one of them could have achieved separately" [8].

Foundation Structures in OSD

A project with a strong champion is not without its own challenges. When one company is the sole champion, especially one that maintains tight control of a large project, it creates a potential conflict of interest between the champion and its community. If the project is championed by only one company, commercial competitors may not want to provide resources to the project as they may feel that it only benefits the champion company and their goals for the project will not be addressed. This mentality serves to decrease the credible promise, legitimacy and trust in the project between the community and the champion. In order to mitigate this risk, Weiss states it may be beneficial to all the stakeholders to transfer ownership of the project to an independent foundation as it "creates an arms-length relationship between the project creator and the project" [2]. This allows the collective to pool resources and centralize common functions without creating an unfair advantage to one member. For example, when considering software development road mapping, a champion specializing in a specific space sensor technology may prioritize development on enabling their niche product and delay other requests from contributors that do not enhance their product (e.g. software push/pull requests to enable 3D visualization, machine learning or team management technologies).

The roles of a foundation are to centralize common functions that all members can access and to provide governance over the project [7]. By pooling resources in a foundation, all members could potentially benefit equally from the shared development. The foundations governance relationship is shown in Figure 1.

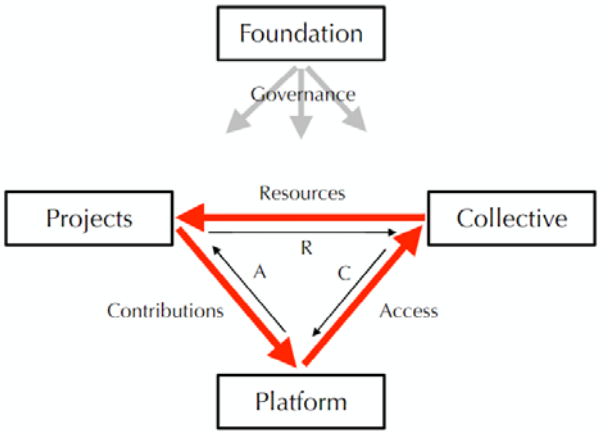

Figure 1 Foundation Governance Relationship. Image sourced from [7] based on [9].

Lynne Markus' "empirical research (on open source) development suggested that the elements of (OSD) governance could be grouped into at least the following six categories of formal and informal structures and rules:

i) Ownership of assets—this category includes intellectual property licenses and formal legal organizational structures (e.g., foundations);
ii) Chartering the project—this term refers to statements of vision about the goals of the project, what the software product should look like, etc;
iii) Community management—this category involves rules about who can be members, how their identity will be verified, what roles they can play, how they can change roles, etc;
iv) Software development processes—in this category are structures and rules that address the important operational tasks of development, such as requirements elicitation, assignment of people to tasks, processes for managing software changes, release control, etc;
v) Conflict resolution and rule changing—this category involves rules and procedures for resolving conflict and for creating new rules; and
vi) Use of information and tools—in this category are rules about how information will be communicated and managed and how tools and repositories will be used" [10].

The reader of this paper may be envisioning what a potential foundation would be, its governance, and champions in the project. This view is based on the readers current environment and vantage point. A reader in a large organization or company may envision a foundation consisting potentially of government and large companies, with small and medium size companies moving up in the foundation as they prove their merit in the project. A consideration in this foundation scenario is that large organizations and companies are usually slow to react and adapt to change, especially change that affects their business and revenue models. In a paper by Chesbrough discussing open source business models and company R&D work, he states that "developing that (open source R&D) capability requires the creation of processes for conducting experiments and for assessing their results. Although that might seem obvious, many companies simply do not have such processes in place. In most organizations, no single person short of the chief executive officer bears responsibility for the business model. Instead, business unit managers (who are usually posted to their jobs for just two to three years) tend to take the business model for granted. For them, running risky experiments in which the payoffs may not emerge for three or more years is not a high priority" [3]. Policies and procedures may not be in place in large organizations and companies to permit participation in a foundation even if there is a strategic advantage to do so. It will take time to develop these items, especially if there is no precedent of open source projects within the organization. If your vantage point is from a

small or medium size company, you may envision partnership with complementary companies to help develop the complete product solution. Small and medium size companies are usually more open to change and new business models. In this scenario, small and medium size companies could collaborate, pool resources, and share the cost of development, enabling them to bring their product to market faster, create standardization/adoption, and develop a more integrated, comprehensive solution. This would potentially allow them to create a value-added product comparable to a large company's in-house solution, permitting these small to medium size companies to compete and disrupt a large company's market place.

In the S-COP case study, a foundation could be created with the champions and stakeholders of the project consisting of commerical companies with a vested interest in leveraging an S-COP platform. Defence organization(s) could also function as champions or advisiors helping guide the project to meet future needs. As the project grows and evolves over time, the members of the foundation may change and potentially include allied Defence organizations, new commerical companies and civilian space organizations.

ARCHITECTURE

Weiss states that "architecture affects the ease with which open source components can be integrated at the Use stage; how contributions to an open source project can be made at the Contribute stage; what parts of the product are under the control of the championing business; and how collaboration is organized at the Collaborate stage" [2]. A type of software architecture that facilitates open source projects is modularity. Weiss' general description of modular architecture is to "partition the code base so that different parts (or modules) can be worked on and managed independently" [7]. This creates an advantage in an OSD project as "code modularity lets many programmers extend the program by working on separate modules, without needing to change or understand the core system, or interfere with each other's progress" [11]. "Without a modular design, it would be extremely difficult to modify the source code of such a large application with so many contributors" [12]. The advantages of a modular architecture include:

i) decreasing the barrier of entry to a project. A contributor does not have knowledge of the entire system, but instead can focus on a specific module in which they have specialized knowledge as modules usually have reduced complexity;
ii) allowing for autonomous development where contributors can work independently and experiment with the modules without impacting the functionality of the system as a whole;
iii) allowing for parallel work. While parallel work is often seen as inefficient, when uncertainty exists in how a solution will be achieved, parallel work allows for experimentation with multiple paths to a solution; and
iv) creating design options that facilitate development of complements (add-ons/plugins), and thus increase the platform's attractiveness [7].

In the S-COP case study, in addition to the advantages stated above, a modular architecture will allow for complements to be added to the base platform to enable additional functionality as required. The base S-COP program, open to the community, would have limited functionality and use unclassified open data streams. This would allow for a more open development as companies could build on this platform. A Defence organization could then add their own modules, complements and data streams operating a different security levels that not available to the public, and country specific. Further analysis is required to determine the best software achitecture for an S-COP program to enable Defence organizations to use while permiting an aspect for open development.

LICENSING STRATEGY

Open source software is distributed with a license that describes the terms and conditions that govern its use [7]. This paper will not provide an in-depth analysis of the different open source licenses, but instead provide a high level overview of how the strategic selection of an open source license(s) can create the desired effect of an open source project. It is important to understand licensing strategy as "licensing issues cut across all levels of the open source engagement model. It affects decisions which components to integrate at the Use stage; the ownership of contributions (Contribute); business strategy (Champion); and governance (Collaborate)" [2]. In Figure 2, Daffara "illustrates how popular licenses may be combined. An arrow from one box to another indicates that those two licenses can be combined and that the combined result effectively has the result of the license at the arrow's destination. To determine whether two licenses can be combined, find a common license that can be reached by pathways leading from each license. For example, an Apache 2.0 license and a GPL2+ license can be combined using GPL3 or GPL3+" [13]. "The choice of an open source license for a project's code base is not clear-cut and depends on several factors. In general, when reusing code that comes from external projects, license compatibility is the major consideration in selecting a license" [13].

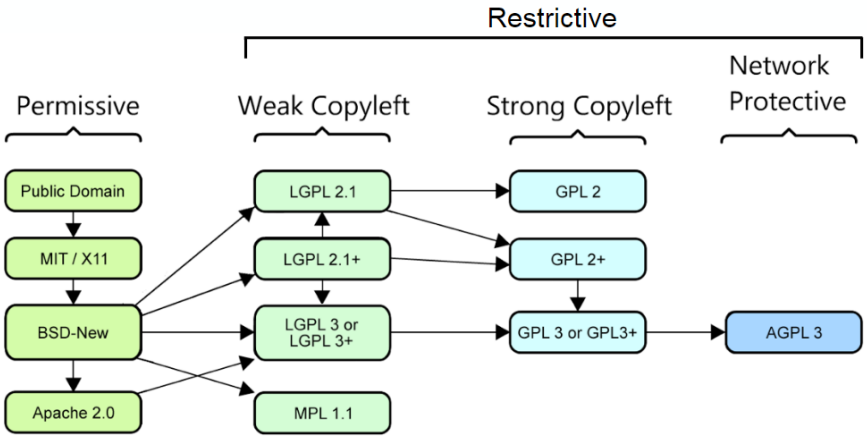

Figure 2 Compatibility relationships between popular open source Licenses. Image sourced from [13].

A permissive license, such as BSD, aids in creating a standard industry adoption of the project [7]. While a restrictive license, such as GPL, is more applicable for projects targeting end users and administrators [7]. It is important to note that the owner(s) of a project may consider using a dual licensing strategy, where a company offering a product (e.g. a library) may offer one version under the GPL (open source license) and sell another under a commercial license to system integrators. Under this strategy, the "buyers of the commercial license are released from some of the obligations of the open source license" [7] such as the requirement to release code changes to back the open source community. When considering dual licensing strategy, Daffara warns that "for contributions to become part of the product source code, external contributors need to sign off their rights to the code so that the company can sell the enterprise version alongside the open version. Note that dual licensing also requires at least one of the licenses to be a strong copyleft license, like the GPL" [13]. The optimal licensing strategy influences the project goals and the degree of adoption by commercial companies as an embedded element [13].

When exploring the open source licensing strategy in the S-COP project, it raises a number of concerns that will have to be addressed. Open source licenses guarantee the following basic rights; i) use the program; ii) access the source code; iii) modify the source code; iv)

redistribute the program in its original or modified form [7]. Open source licenses and projects are designed to foster an open community and pool resources. In defence applications, additional considerations (such as controlled goods and information/data dissemination restrictions) may require the community participation to be controlled and therefore open source licenses do not meet these requirements. When asking the question of what has to be controlled, the easy answer is that everything regarding the military and space has to be controlled and protected. To debate this point, it is important to analyse what products are currently being employed in the operational role and if they require a controlled approach. Looking at space operations, we see that a number of software products are already open source (propagators or programing languages such as Python and its libraries) or COTS (e.g. MATLAB or STK) and can be used/purchased by the public. While there are a number of protected data streams, there also exist open data products such as a two line element database. This brings up the discussion of when there needs to be control, if it's the data, the software/algorithms, or the situational knowledge and tactics of the operator. This discussion is outside the scope of this report, but the answer is complex and depends on the item being discussed.

It is important to understand these concerns as it will affect how the community participates and the governance structure of the project and how the community interacts with it. If S-COP could be viewed as a platform of basic common functionality with modular architecture, that companies could build applications on, analogies can be drawn to commercial examples such as the Android operating system with its app store. The champion/owner/ collective/foundation could own all of the core code intellectual property (IP) or release a portion to open source. Other organizations could build and sell components upon the S-COP platform with a SDK, which allows them to choose the protection of their intellectual property.

A dual licensing strategy could be used for Defence organizations to have a commercial license of the S-COP platform, negating the requirement to publish their modifications of the code base, and through modular architecture have modules to plugin to the S-COP platform (e.g. protected data streams or applications) that are not accessible to the public. This discussion really depends on if a base S-COP platform could exist and thrive in the open, outside of a Defence organization. Could an S-COP platform be used by civilian space agencies, commercial companies, and academia/R&D? If the answer is yes, it may be worth exploring what could be made open to foster the community. If the answer is no, then the focus should shift to more control and commercial agreements between organizations for collaborative development.

Export and Import Permits Act administered by the Trade Controls Bureau of Global Affairs Canada [14] must be addressed when considering potential application in a Defence organization. Representatives authorized to advise on this act should be consulted as early as possible in project formulation by the Champion. The Champion as part of their role, should then provide guidance documentation to educate the community to proactively recognise and address the Controlled Goods requirements.

CONCLUSION

This paper opened with the understanding that Defence organizations "must have a software environment that is easily adaptable to changing mission needs; this software must also evolve at lower cost and be delivered rapidly so it can be used when it is needed" [1]. This paper examined the framework of open source development to the DRDC Ottawa S-COP project to determine if open source and collaborative development methods could be used as a development approach. Open source and collaborative development methods were discussed as potential solutions to address these requirements, but requires special considerations to be

applied to a defence organizational ecosystem. As this paper only provides an introductory overview of the topic, further exploration of each engagement stream is required.

ACKNOWLEDGEMENTS

The author would like to acknowledge Dr Julie Lefebvre, Director General Joint Force Development and ADM S&T for providing support for the Space Common Operating Picture effort.

REFERENCES

[1] J. Scott, D. Wheeler, M. Lucas and J. Herz, "Software is a Renewable Military Resource," *SoftwareTech,* vol. 14, no. 1, pp. 4-8, 2011.

[2] M. Weiss, "Business of Open Source," *EuroPloP,* 4-8 July 2018.

[3] H. W. Chesbrough , "Why Companies Should HaveOpen Business Models," *MIT Sloan Management Review,* vol. 48, no. 2, pp. 22-28, 2007.

[4] Matthew Langham, Indiginox, "The Business Of Open Source," *OSSWATCH,* 3 February 2009.

[5] M. Weiss, "Economics of software product development collectives," *Technology Innovation Management Review,* vol. 1, no. 1, pp. 13-18, 2011b.

[6] F. Dixon, "Lessons from an Open Source Business," May 2011. [Online]. Available: https://timreview.ca/article/441.

[7] M. Weiss, *Open Source Business,* Ottawa: Carleton University, 2019.

[8] J. Scott, D. Wheeler, M. Lucas and J. Herz, "Running Open Technology Development Projects," *SoftwareTech,* vol. 14, no. 1, pp. 26-31, 2011.

[9] S. Muegge, "Business Ecosystems as Institutions of Participation: A Systems Perspective on Community-Developed Platforms," *Technology Innovation Management Review,* pp. 4-13, November 2011.

[10] M. L. Markus, "The governance of free/open source software projects: monolithic, multidimensional, or configurational?," *Journal of Management & Governance,* vol. 11, no. 2, pp. 151-163, May 2007.

[11] M. Aberdour, "Achieving Quality in Open-Source Software," *IEEE Software,* vol. 24, no. 1, pp. 58-64, 8 January 2007.

[12] M. Kennedy, "Evaluating Open Source Software," vol. 44, no. 1, pp. 12-15, 2011.

[13] C. Daffara, "Open Source License Selection in Relation to Business Models," *Technology Innovation Management Review,* February 2011.

[14] Global Affairs Canada, "https://www.international.gc.ca/," Decemeber 2015. [Online]. Available: https://www.international.gc.ca/controls-controles/about-a_propos/expor/guide-2015_toc-tdm.aspx?lang=eng. [Accessed April 2019].