A System of Systems perspective on Open Source Software Projects

Mr. James Cowling[1] and Dr. Robert Cloutier[2]

[1] Stevens Institute of Technology, USA, jcowling@stevens.edu
[2] Stevens Institute of Technology, USA, robert.cloutier@stevens.edu

Abstract
In recent times new forms of development enterprise have come to prominence, notably open source software projects (OSSPs), that appear to be difficult to successfully recreate. This paper applies a System of Systems (SoS) perspective to bring to the fore a possible framework for underpinning the establishment of OSSPs. A SoS emerges from a collection of diverse highly inter-connected yet autonomous component systems, that participate to further their own and SoS objectives [1]. Comparing and contrasting SoS characteristics with the essential features of OSSPs, namely decentralized governance, community diversity and evolutionary development, offers the potential for deeper understanding of OSSPs. Viewing OSSPs as the combination of independent development efforts shifts the focus from observing the behaviour of the OSSP as a whole to that of the interactions of a community of contributors. By considering OSSPs in this way it may be possible to develop a conceptual framework. Such a conceptual framework would capture the fabric of these projects, offering a potential lever for their successful formation.
Key words – Open Source Software Projects, System of Systems, Governance, Framework

1 Introduction

Today's enterprises are increasingly interconnected and rely on a wide range of interrelated systems of varying provenance. These "legacy" systems are modified and integrated together with new capabilities over time to form Systems of Systems (SoS) in the support of enterprise objectives. These SoS have been described with a set of characteristics that distinguish them from 'mere systems' [1] and suggest traditional System Engineering approaches are inappropriate for their creation.

In recent times new forms of system development have come to prominence, particularly open system development approaches. The bane of adopting the open system development approach is the apparent difficulty with which projects based on this method are recreated successfully. The reason that success is illusive may be because of an inability to grasp the essential features of the development system itself and hampered by a lack of a sufficient framework for its understanding.

System Engineering is traditionally viewed as a rather linear sequence of capturing and describing requirements that leads to the development of an architecture, designs, and concludes with construction of system components. Once the components of the system have been created a similarly linear process of integration and testing concludes with user acceptance of the complete system. Open system developments such as Open Source Software Projects (OSSPs) diverge significantly from such an approach in the way they create software applications and in the establishment of the OSSP itself. The methods for establishing successful OSSPs are likely to remain illusive if we apply a 'mere system' model to its understanding.

However, OSSPs appear to share similarities with characteristics associated with SoS, the investigation of which may begin to reveal the conceptual framework underlying open system developments. The intent of this research is the investigation to determine whether a more detailed understanding of the fundamentals of the open system development approach may in turn inform the development of a methodology for SoS engineering.

2 The System of Systems Concept

Taken together, five characteristics allow us to describe a SoS as collection of highly inter-connected component systems, of diverse capability, each with a high degree of autonomy, that invest in the system, which emerges from the combination of those components, to further their own and the system's objectives. [1]

2.1 Autonomy

A system is brought into existence to perform some function and is essentially 'free' to fulfil that function, defining autonomously how it will pursue its purpose. The same is not true of the parts that make up a system. Parts are selected to meet the needs of the system, and are only included in the system if the system deems them necessary to the performance of its function. The part is given purpose by its inclusion in the system, on it's own, outside the context of a system it serves no purpose. [1]

2.2 Belonging

While the components in a SoS display a high degree of autonomy they are not completely autonomous. The SoS makes demands of them such that it is necessary that the component system sacrifice some of its ability for self-determination, at least in terms of how it contributes to the SoS. Clearly the SoS is also modified by the inclusion of the component system and this conjoining happens with consideration and negotiation. But one must not forget that the SoS represents a collection of component systems and therefore the negotiation takes place between each of the constituents. The negotiation that takes place when a component system joins the SoS centres on the cost (relinquishment of self determination) each component has to pay for the benefit to their own purpose and that realized by the resulting SoS. Each component system therefore determines the return on investment that would be gained by belonging to the SoS. [1]

2.3 Connectivity

When designing a system its parts and their connectivity are considered simultaneously, but this is not possible for a SoS. The constituent systems already have their connections defined and yet it is necessary for them to support interoperability with other, previously unassociated systems. New connections must therefore be supported that imply an altered boundary for each of the constituent systems. The systems, displaying both autonomy and belonging, have a decision to make in respect of which connections they will support to enable the achievement of SoS (and their own) objectives. [1]

2.4 Diversity

A SoS should have much higher diversity amongst its constituent systems compared to that of the parts, which have limited diversity by design, within a mere system. Diversity allows great adaptability, in the face of new environments and capability needs, for the SoS. Increased diversity is achieved by enabling autonomous systems to join together through open connections to form the SoS. [1]

2.5 Emergence

In coming together the whole is greater than the sum of the parts. The collection of constituent systems does not constitute the construction of the SoS that is brought into existence. It is during the process of gathering together that the SoS structure is created. The relationships between constituent systems are defined by cost/benefit trade offs each makes autonomously in deciding whether to belong to the SoS. The resultant connectivity within and capability of the SoS is not necessarily predictable as it is dependant on the relationships the decisions the constituent system make. [1]

3 Open Source Software Projects

The open source software movement that has emerged and grown over the last decade or more may now represent a viable alternate to prevalent commercial software practices. Although it is often presented as offering the potential of producing better software, quickly and cheaply, OSSPs are fundamentally different in both organization and development process [3], [5].

OSSPs are resourced by potentially large numbers of geographically dispersed, predominantly volunteer contributors who select for themselves the task they will perform based largely on their own interests and skills [3], [5]. Developers are attracted to join the community by a complex collection of motives including creating needed features, learning new skills, philosophical beliefs, enjoyment of freedom to be creative, and sometimes political statements about non-open practices [7].

3.1 OSSP Development System Lifecycle

Early in the OSSPs lifecycle a relatively small community (maybe an individual) begins developing code. Depending on the size and pedigree of this early developer group project structures maybe agreed, however, more than likely they emerge over time as the project grows. Once the project has been successfully initiated, sometimes signalled by a first stable release of code, it enters a period of growth as it attracts greater numbers of contributors until it reaches it's maximum size. The growth phase is often followed by a period of decline in which developers leave the community. The decline in numbers of contributors can result in the collapse of the project with too few to sustain development. Alternatively, the project may undergo a revival and enter a period of sustained improvement. [4]

3.2 OSSP Governance

The manner in which a project is governed typically develops as the size of the community expands. While the project remains relatively small it is easy for decisions to be made and consensus to be reached, but this does not scale well and growing projects are therefore driven to establish decision mechanisms. Over time most project governance has been seen to mature from initially resting with its members to the emergence of meritocracies and voting structures [7]. An open source approach is critically dependant on allowing all points of view to be plausible in pursing a course of action and to this end great care is required to ensure the formalization of governance structures do not undermine the projects "openness" [7]. Structures that are designed to maintain the position of long-standing members over newer, regardless of competence, may result in preventing the introduction of new ideas and reduce the projects likelihood of entering revival phase of its lifecycle.

In open source projects a minority of the community produce the majority of the code and are likely to emerge as the community leaders during the early phases of the project [2]. As the project grows not only do the contributions of more developers need to be coordinated but also the organizational complexity increases as it is divided into 'manageable' size modules each with it's own responsible sub-group [5]. As the project structure becomes more complex procedures become formalized and the early leaders could find themselves at the head of a hierarchy [2]. As For this hierarchy to not damage the project it must preserve the plausibility of each contribution [6]. The maturing of successful open source project's governance has been seen as a move from direct, where decisions are shared equally amongst the community, to representative democracy [7].

3.3 Contributors Motivations

The roles undertaken by the participants in an OSSP can be characterised by the nature of their contributions. "Bug fixers" who form the vast majority of a project community, identify bugs and occasionally propose patches are extrinsically motivated (they desire working code) [4], [7]. "Programmers", a much smaller, highly contributory group are more tightly integrated into the project governance structure [4]. Programmers are either motivated by improved personal reputation, optimization of the code or both and therefore display a mix of intrinsic and extrinsic motivations [4], [7]. The projects founders or the most experienced programmers become the project's "leaders", the smallest proportion of the community [4]. Leaders take responsibility for strategic focus of the project and head the governance structures [4].

Bug fixer's and programmer's motivations remain fairly constant throughout the lifecycle of a project, however those of the leaders change as the project progresses [4]. Initially, the founders are mostly intrinsically motivated but as the project grows those in leadership position become more extrinsically focused. By the time the project reaches maximum size the leaders are almost entirely extrinsically motivated [4]. The shift from intrinsic to extrinsic focus seems entirely consistent with a need to create a vision with which to establish and grow the project. However, after maturing to its largest size, if the fall in numbers is not to result in collapse the leadership must again become intrinsically motivated [4].

3.4 OSSP Infrastructure

OSSPs use none of the coordination mechanisms associated with more traditional approaches such as a schedule, or architecture [5]. Individuals self assign themselves to develop functionality thy feel is required and contribute on a voluntary basis, without guidelines regarding time and intensity of work. As such there is typically no project manager, product breakdown structure or project plans. [4], [3]

The lack of prescriptive architecture or project plan may enable OSSPs to readily adapt to new innovation or more optimally functional software. However, without these coordination tools participants must be able to communicate readily at low cost in terms of the expense and effort required. Increasing the diversity of the contributors by reducing the barriers to participation further enhances this adaptability. As a result most open source software projects utilize a limited set of common tools and readily accessible, Internet based asynchronous communications mechanisms (e.g. mailing lists) [3], [4].

3.5 OSSP Processes

Lack of a predetermined set of requirements prevents the development of an architecture in the traditional sense and often there is no explicit system-level design, or detailed design that the developers work to. OSSPs instead rely on the community exposing requirements as a result of previous developments, either in the form of indentifying bugs or opportunities for additional functionality. Once requirements are identified it is the willingness of the programmers to develop a solution that drives the OSSP change control process. [3]

While OSSPs do not create an architecture per-se, the fact that they are typically a collection of highly distributed developments does necessitate certain architectural principles to ease integration. Most prevalent of these principles is that of modularity required to allow the combining of components developed separately and new features to be added as new requirements are identified. [5]

Many developers produce solutions to the requirements they wish to work on and the design emerges as a result of the best combination (as determined by the community) over time. In essence many competing solutions are developed, integrated together and a process of natural selection takes place. [5]

Periodically the best code and new functions will be collated into an "official" release that is made widely available for distribution. During this process new functionality is only included in a release if it works with the rest of the application. Code is rarely modified during the integration phase to make it fit into the application and this reinforces the modularity principle. Once the release is made available to the community the next round of improvements begins with exposed bugs and functionality requirements. [5]

This iterative development and improvement based on community selection of optimal code modules and operational usage results in the software "evolving" from one release to the next.

4 Essential OSSP Features

The critical indicators of a project's openness appear to be the establishment of decentralized democratic structures during the project's growth phase (as opposed to the formalization of a hierarchy to reinforce individual's positions), a highly diverse community (may have the potential to create a greater number of viable alternatives to sustain an evolutionary development than one with low diversity), and a development process (requirements, release management, etc.) that supports an evolutionary/iterative application lifecycle (that supports the 'natural selection' of successive optimal solutions as opposed to a planned or prescriptive linear lifecycle).

The volunteer status of contributors is often highlighted as a feature of OSSPs, however, many leading projects do have a paid staff (e.g. Mozilla is supported by the Mozilla Foundation). It may be that to be successful a proportion of participants are employed purposefully to contribute to the project. For the purposes of this paper the effects of volunteerism have not been considered an essential feature of OSSPs, although it is recognised as a potentially contributory factor to achieving high diversity.

The majority of projects hosted on sourceforge.net appear dormant while a few prosper; it would appear the infrastructure that supports both is not the distinguishing feature and therefore is not considered further here. While the infrastructure and tools that connect participants are important to any open system, this paper proposes that it is the nature of the interactions themselves that is vital.

4.1 Governance – Decentralized decision making

Most open source projects have a meritocracy of developers that emerges based on competence, experience and community respect [2], [7]. Often the most influential developers in a project are those who joined during its initiation and where subsequently "elevated" as the project grew and governance structures where established [2], [7]. However, to remain a viable OSSP the project governance mechanisms must allow and recognize contributions from the whole community [6]. The approach the leaders take in setting strategic direction for the project may either lead to a move away from this inclusive approach or facilitate a more open, decentralized organization. Therefore the most open organizations are those that achieve maximum decentralization. This decentralization enables greater optimization at various levels within the project and is therefore also a factor in the projects ability to evolve and innovate.

4.2 Community – An "organic" community

A non-organic project is one in which most of its members (particularly leaders) are affiliated with each other (outside of project activities) through some external entity and seek to control the project to the benefit of that entity. For example, the Open Solaris project has been criticized because of the highly influential Sun employed developers that are perceived to further only Sun's objectives, presumably to the perceived detriment of the rest of the community. For the purposes of this research it is proposed that an organic project is one in which no external affiliation attempts to exert a controlling influence beyond a desire to see the project succeed. Infrastructure, project complexity, technical skill level, and specialism maybe factors that effect diversity of a project but efforts should be made to limit unnecessary bars to entry into the developer community. Greater diversity also increases the potential for innovation.

4.3 Development – Iterative, evolutionary processes

Requirements management, sustained product improvement and release control are factors in determining a projects evolutionary nature. In the most open of projects a requirement can originate from any community member, receive mutual agreement (or rejection) by the community and can be satisfied by any developer [3], [5]. Adherence to component structures and interfaces are encouraged to aid incremental development and release configuration [5]. For the resultant application to be viable for users it must remain 'healthy' supporting the rapid resolution of bugs, adapt to new uses and operating environments. This culminates in release management process that delivers new versions at a rate that keeps pace with users changing needs.

5 Applying the SoS concept to OSSPs

Applying the SoS concept to OSSPs is a novel approach and will be explored next. If one considers an OSSP to be a network of contributors, each of wide ranging software development motivations, that under take coding tasks of their own devising to further their interests that are, to some degree, in common with those of the community. It is then possible to describe OSSP essential features in terms of the of the SoS characteristics (as shown in table 1). Subsequently it is possible to create a definition of subtly different emphasis where the OSSP emerges as a consequence of the contributor's interactions as follows:

OSSP participants enjoy the freedom of self-tasking and contribution determination which, when combined with those of other independent participants, fuels the development process and determines the structure of the project. By exercising self and social controls, contributors aim to comply with and sustain the project with a culture that will secure their return on investment, gaining a highly functional application they may not have been able to create themselves. Although highly respected members serve to sustain strategic direction it is the diverse range of viewpoints held by the community members that creates innovative variety within the project. While the act of code creation remains a largely solitary pursuit the well-connected developers share experiences and results for the benefit of the entire community, enabling selection of the best of solutions as the resulting application evolves.

By considering an OSSP in this way the focus is shifted from observing the behaviour of the project as a whole to that of the interactions of a community of individual contributors. It is much clearer in this description that project characteristics emerge when the community comes together and as a result of that togetherness. Thus it is not the OSSP structure that allows individuals the freedom they enjoy; rather contributors exercising their freedom create the project structure. Put differently, the OSSP exists as a result of the contributor's interactions not as a precondition to facilitate them. This feels somehow obvious or natural. Considering the OSSP as a 'mere system' may have clouded observers by encouraging thoughts of project structure design as opposed to the emergence of complexity from the simple interactions of participants.

Table 1 – Application of SoS Characteristics to the OSSP Essential Features

	Governance	Community	Development
Autonomy	The community creates a culture which requires self controls that are reflected, sustained, reinforced and exercised by social controls	Each contributor undertakes activities that suit their skills, interests and satisfies their own motives	Multiple self tasking contributors creating multiple variations and versions of software functionality is necessary for the OSSP development process itself
Belonging	Seniors in the meritocracy provide strategic direction for the OSSP	By giving time, effort and expertise to the OSSP contributors gain access to collective support of the community and breadth of functionality they may not otherwise	Contributors join the OSSP to be, fundamentally, involved in the creation of freely available software functionality
Connectivity	Rapid and wide exchange of ideas, experience and software across the community enables project decision making	Critical to cohesiveness is community wide communication that establishes a shared purpose	Continual reconfiguration and integration of new functionality engenders collective knowledge in each of the contributors
Diversity	A wide variety of view points drives the exploration of the problem space and the creation of plausible solutions from which to select	A broad range of participants assists innovation and variety of functionality created	An extensive collection of plausible solutions and reviewers underpin the OSSP development process
Emergence	Larger, successful OSSPs tend to develop, over time, a form of representative democracy for decision making	The community changes over the OSSP lifecycle – the community changes the OSSP objectives while the functionality needs attracts/dissuades contributors – hitting a 'sweet spot' is required for sustained productivity	Community decision of the best fit functionality, over time, establishes an evolutionary development system – 'natural selection'

6 Establishing an OSSP

When systems are constructed it is typical to think of the system level characteristics of that system, the functions it will perform and its proficiency, before, or at least at the same time as consideration is given to the inner workings of its components. Occasionally systems are constructed from available components but even then they are assembled with a vision of the resulting system in mind. In either of these bottom up or top down system development approaches the engineer has control over how the components will be brought together to form the system. This is not the case with the emergence of the OSSP development system.

For the OSSP it is the community members that define the functionality and performance of the development system, not by deliberate design and recruitment but merely through their participation in project's software development. If the community where to attempt to design and construct their system upfront the very thing they we striving for could be lost as it can only emerge through participation. Focusing on the project structure rather than creating conditions conducive to participation may be one reason open system developments appear so hard to re-create. Conversely, it is difficult to see how a fledgling community forms without some concept of structure to form around.

To seek a resolution to this conundrum we need to return to the earlier description of an OSSP lifecycle and consider the initiation stage of the development system. OSSPs typically start either by a very small group, possibly a single developer, or in a closed development system such as a corporate software house. For the single developer the development process maybe more ad hoc and for the corporation much more regimented, but nevertheless in either case the development system likely has a determined structure in this 'pre-community' stage. The challenge, therefore, could be in transitioning from the 'closed' to 'open' states.

The danger is for the original development system to "feel" that others are joining it as opposed to the 'original' and 'new' contributors joining with each other to form a new development system. Although a project that maintains the original development system may achieve some success in the short term, over time, not moving to a more open construct may potentially prevent the project entering the desired sustained improvement state.

This is not to say that contributors actively renegotiate the OSSP development system each time the membership changes, the chances are they are unaware of a structural change occurring. There is also a dampening effect as those joining are likely to comply, to some degree and at least initially with the existing norms. Therefore any changes new members may bring could be subtle or exercised over an extended time as they become more established within the community. The impact of membership change may be more apparent in small projects, but in general it is possible that each contributor can only perceive their interaction with the community and has no macro view of the community structure.

7 Engineering an OSSP

By applying the SoS concept we can see that the development system that is the OSSP has to be allowed to emerge, not only from its (potentially) closed form but also from the community itself. The OSSP, in some sense, cannot be engineered directly if it is to emerge, however conscious effort is required if the emergent system is not to return to the apparent predictability of a less open approach. By considering the OSSP as a collection of contributors and focusing on how those contributors participate to form the project it may be possible to develop a conceptual framework. The conceptual framework would describe 'the space between' the contributors, the fabric and essence of the project. Such a framework could offer the potential lever for shaping a project and the promise of an engineering mechanism.

7.1 A framework for OSSPs

The framework should not be a set of rules. Rules will be broken so need adjudication, which potentially leads to the creation of a hierarchy and centralization. Once the project governance structure becomes centralized it is no longer an OSSP, according to the essential features, and may have a tendency toward collapse in the longer term. Instead the framework could be more usefully thought of as guiding principles that the community aspire to.

Guiding principles for an OSSP could be based on or derived from the essential features introduced above and potentially expressed in the following ways:

- The community, particularly leaders in the early stages should resist to the formalisation of a hierarchy or adoption of command and control structures. Meritocracy should give way to representative democracy.
- The community should enable, certainly not prevent a variety of leaders to come to prominence and take the various aspects of the project forward
- Self-controls should be encouraged and social controls relied upon to not only create a more cohesive project but one that is able to respond to disruptive behaviour quickly without the delay or potentially divisive adjudication process
- Entry and "acceptance" into the community should be as constraint free as possible to encourage the greatest possible diversity
- The principle of self-selecting code development task, and iterative product release must be preserved

7.2 Defining a framework

The need for a framework is somewhat paradoxical. A young, small project (say of three contributors) is unlikely to benefit from having a framework instead relying on more ad hoc structure that suits all the contributors. Likewise a mature, large project (of 3000 or more) is of sufficient capacity that the principles of openness are engrained and self-sustaining such that the burden of consciously maintaining a framework may become divisive.

The critical stage for an OSSP appears to be the period of decline it goes through prior to either revival or collapse. It could be that a factor determining the achievement of sustained improvement state and collapse is the degree with which openness has taken root at this point. Either way, it is probably too late to begin framework establishment during the decline phase, the likely tendency would be towards a "who's going to do what" mentality in the face of falling numbers. This could result in a move further away from openness and increase the likelihood of collapse. Much better to have the framework in place so that 'closed' principles that could condemn the project to collapse, or at least hinder its recovery, are avoided.

The period during which the project grows to its maximum size therefore appears to be the most opportune time for developing the framework and, if purposeful engineering of an OSSP is possible, the most effective. During this period thoughts are likely to be of growing the community to increase code production and diversify solutions ultimately leading to improved functionality. In essence open qualities are likely to be most highly valued during this period, and a framework that enhances those qualities is more likely to be accepted.

8 Conclusion

Rather than try and discern the structural characteristics of successful projects, developing an understanding and classifying the characteristics of each contributor's participation within a successful project may be more fruitful. The structure of a project may be dependant on the field of endeavour, problem space and the community, where as the framework underpinning successful projects that have emergent structures is potentially much more transferable.

It is unlikely that project's frameworks are written down, rather it is held in the collective perceptions of the community, so identifying the characteristics of successful frameworks may be challenging. Nevertheless, the impact of the framework that is established during its growth period on the project's ultimate collapse or revival seems worthy of further research.

9 References

[1] Boardman J.T., Sauser B.J. (2006), Systems of Systems – the meaning of of, Proceedings of the 2006 IFF/SMC International Conference on Systems of Systems Engineering, Los Angeles, CA, USA, April 2006

[2] de Laat, P. B. (2007), Governance of open source software: state of the art Journal of Management & Governance, v 11 no 2, pg 165-177

[3] Gurbani, V. K., Garvert, A., Herbsleb, J. D. (2006), A case study of a corporate open source development model, Proceedings - International Conference on Software Engineering 2006, p 472-481

[4] Lattemann, C., Stieglitz, S. (2005), Framework for Governance in Open Source Communities; Proceedings of the 38th Hawaii International Conference on System Sciences 2005

[5] Mockus, A., Fielding, R. T., Herbsleb, J. D. (2002),Two case studies of open source software development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology, v 11, p 309-346

[6] O'Mahony, S., Ferraro, F. (2007), The Emergence of Governance in an Open Source Community, Academy of Management Journal; v 50 no. 5, pg. 1079-1106

[7] Shah, S. K. (2006), Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development, Management Science; vol 52 no. 7; pg. 1000-1014