Entropy-Based Study of Components in Open-Source Software Ecosystems

LIGUO YU[1], JOHN CAWLEY[1], SRINI RAMASWAMY[2]

[1]Computer Science and Informatics
Indiana University South Bend
South Bend, IN 46634, USA
ligyu@iusb.edu; jjcawley@iusb.edu

[2]Industrial Software Systems
ABB Corporate Research Center
Bangalore, India
srini@ieee.org

Abstract. Open-source software products contain free components that can be used by both individuals and companies. When open-source commercial off the shelf (COTS) components become integral parts of a commercial product, the open-source project and the commercial software organizations tend to form a relationship akin to an ecosystem. In this paper, we apply entropy to measure the diversity of an open-source ecosystem, in which it represents and measures the attractiveness of open-source components. We studied 121 Eclipse products to understand their evolution. Our study finds that although most of the Eclipse products' popularity increase over time, their representative product, the Java IDE's popularity has remained relatively stable in recent years.

Keywords: entropy, software component, open-source software ecosystems.

(Received January 23rd, 2012 / Accepted March 15th, 2012)

1 Introduction

Software products are, in general, not used in isolation. There exist some interdependencies among related software products, which compromise the base for large-scale complex software systems. For example, at the macro level, application software depends on the underlying system software; any changes to the system software might affect the application software which is built upon it. On the other hand at a micro level, large software systems are compositions of several software components; advances in some of the software components might force the need for an upgrade of the host product in order for companies to keep (or gain) market share. Therefore, software products and their dependents can be thought of as forming an ecosystem.

The study of complex software system as an ecosystem of sorts has been gaining increased attention over recent years. Yu et al. studied the co-evolution of open-source products and their relations with commercial software companies [33, 34]. In their representative book [24], Messerschmitt and Szyperski, studied the concept of software ecosystem, which includes users, developers, sellers, society, government, and economics. Boucharas et al. presented a network modeling based approach to study software ecosystems [7]. In their series publications, Bosch et al. used the concept of ecosystem in the study of software product line and their relation to global software development [4, 5, 6]. Berra and Rapicault studied the dependency management of Eclipse p2 project, where plugins are considered as the species in an ecosystem [2]. However, compared to other fields, the study of soft-

ware as an ecosystem is still a nascent research area. One possible hurdle that scares researchers is its inherent interdisciplinary nature: software ecosystems involve many subjects from different fields; including software engineering, market analysis, symbiosis, and social networks. Therefore, a well proved domain specific method might not solve the problem of understanding software ecosystems. What we need are interdisciplinary approaches that can solve problems which transcend specific domains.

In this study, we apply entropy, an interdisciplinary concept to study software ecosystems, formed by open-source software components and software organizations, where it is used to measure the popularity of eclipse components in commercial software companies. For completeness, entropy is a commonly used concept in many disciplines; for example, in thermodynamics it is used to measure the randomness of particles, or in information theory, it is used as a measure of the amount of information contained in a program unit. In this paper, the concept of entropy has been applied to analyze the evolution of the popularity and diversity of Eclipse open-source software ecosystems, which contain 121 products and 87 software organizations.

To the best of our knowledge, entropy has not yet been applied to the study of a software ecosystem, which can be viewed as being similar to the measurement of the existence of order or disorder among the participating software components, software products, or software organizations. Because of the ability to study this property in such large-scale complex systems, wherein software components are often distributed and contextually integrated, through which, software organizations are often loosely coupled, we have applied this idea here. Insights gained from this exploratory research will be valuable for organizations that use OSS software components in product development process, or even integrate portions of such components and package them into their core products.

The rest of this paper is organized as follows. In Section 2, we review the concept of entropy and its recent applications within the software engineering field. Section 3 describes related work of applying the entropy concept in software ecosystems. Section 4 presents our case study on Eclipse software components. Threats to validity are discussed in Section 5. Our conclusions and observations are detailed in Section 6.

2 Related Work

The concept of entropy originated in thermodynamics [22], where it is still used to measure disorder amongst particles; such as atoms, molecules, plasma, in a closed system. Later, entropy has been extended to information theory, where it has been used to measure the amount of information contained in a program unit, such as variable, expression, function, and query [29]. In recent years, the application of entropy has been extended to market analysis [12, 27, 23]; information security [9, 1, 26], etc.

The concept of software entropy is defined by Jacobson et al. as follows [17].

"*The second law of thermodynamics, in principle, states that a closed system's disorder cannot be reduced, it can only remain unchanged or increase. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, always increases. This is known as software entropy*"

In this section, we review related work in applying entropy in the study of software engineering issues - such entropy-based work have been diverse, including: complexity entropy, language entropy, author entropy, etc.

In the area of software development and evolution, Lehman has used the concept of software entropy to describe the evolution of software complexity [21]. Specifically, the Lehman's laws of software evolution state that software evolution is inevitable and its complexity will increase during the evolution process unless restructuring is performed. Therefore, reducing software entropy, which can accordingly reducing software complexity, is an important task in software development and evolution. For example, Bianchi studied software degradation from the view point of entropy [3]. Hunt and Thomas used Fixing Broken Windows as a metaphor to reduce entropy in software systems [16].

Hassan and Holt [15] have used entropy to represent the complexity of software development processes. They studied the source control repository of six open software projects and found that complexity entropy is a good indicator of software quality [14]. Jos et al. proposed to improve software process using an entropy-based approach [18]. Hanssen et al. studied entropy in agile software development [13]. Other work in this area includes Olague et al. used the concept of entropy to assess object-oriented programs [25], and Jung et al. used entropy to design metrics for web applications [19].

Entropy has also been used in other areas of software engineering. For example, using the concept of entropy, Yu et al. studied the diversity of software market shares [35]. Specifically, they studied the evolution of software product diversity, such as program-

ming languages, operating systems, web browsers, and web servers. Krein et al. proposed the concept of language entropy, which represents the diversity of programming languages used in a software project [20]. Their study found that language entropy is strongly correlated with the size of monthly project contributions. Taylor et al. defined author entropy and used it to represent the authorship diversity of software program [31]. They found a bimodal distribution of entropy for files with two authors and unimodal distribution of entropy for files with three or more authors. In a further study, they found that when two authors contribute to a file, larger files are more likely to have a dominant author than smaller files [8].

Other applications of entropy in software engineering fields include metric design [28], risk assessment [30], and measuring software structure heterogeneity [32].

In summary, intensive work of applying the concept of entropy in the software engineering field has been performed. However, no work of using entropy to study the software component-based ecosystem has been performed. The objective of this study is to fulfill the gaps in this area.

3 Software Ecosystem Entropy

In this study, we considered a closed ecosystem formed by an open-source product (component) and its dependent commercial companies. Assume an open-source product (component) OSP, which is sponsored by n commercial companies, $C_1, C_2, \ldots, C_n$, each with percentage contribution $P_1, P_2, \ldots, P_n$, respectively. The entropy of this closed software ecosystem CSP is defined as

$$E(\text{CSP}) = -\sum_{i=1}^{n} p_i \log_2 p_i. \quad (1)$$

Equation 1, known as the entropy function, was first introduced by Boltzmann in the 1870s. It is also the general formula for information entropy [29]. The entropy (E) measure in Equation 1 represents the attractiveness and marketplace diversity of an open-source component. For example, if an open-source component is supported by a small number of commercial companies, the entropy (E) value of this open-source component is low, which means the popularity and marketplace diversity of this open-source component is low; if an open-source component is evenly contributed to by large number of different commercial companies, the entropy (E) value of this open-source component is high, which means the popularity and the diversity of its marketplace is high. Note that, if all the n commercial companies of the open-source component have equal contributions, then

$$p_i = \frac{1}{n}, i = 1, 2, 3, \ldots, n. \quad (2)$$

For this case, we can refine Equation 1 as,

$$E(\text{CSP}) = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n \quad (3)$$

Thus, the upper bound for the entropy function is given by Equation 3. Consider an open-source component with four contributing companies, C_1, C_2, C_3, and C_4. Let us examine five different distributions of their contribution schemes (Table 1). (We have previously used a similar analysis method for studying market share [35].) It should be noted here that (1) these schemes are hypothetical and used for explanation purposes only; and (2) the entropy value is calculated using Equation 1.

Table 1: Four commercial companies with five different distributions of contribution schemes and the corresponding software ecosystem entropy

$Scheme$	C_1	C_2	C_3	C_4	$Entropy$
S_1	100%	0	0	0	0.00
S_2	75%	25%	0	0	0.81
S_3	50%	50%	0	0	1.00
S_4	50%	55%	25%	0	1.50
S_5	25%	55%	25%	25%	2.00

In Table 1, Contribution Scheme S_1 has entropy value 0, which indicates that the component is most unpopular and least diverse. In Contribution Scheme S_2, two companies have varying contributions. The entropy value is 0.81, which indicates that the component is more popular and its market is more diverse than Contribution Scheme S_1. In Contribution Scheme S_3, the contributions of companies C_1 and C_2 are more evenly spread than Contribution Scheme S_2, and therefore the system has a greater entropy value (1.0). Similarly, three companies have contributions in Scheme S_4, which means the component is more popular and has a more diverse marketplace. The entropy value is 1.5. In Scheme S_5, four companies make equal contributions and the entropy value is 2, which is the upper bound of a software ecosystem with four commercial companies.

Table 1 shows that the entropy measure of a software ecosystem increases as the popularity and marketplace diversity of the open-source component increases. Therefore it follows that one can use software ecosystem entropy to represent, measure, and study an open-

source component's popularity and diversity in the marketplace. As the components become more diverse and uniformly accepted across multiple constituents, we can potentially infer that the component has reached a level of marketplace stability, i.e. the ecosystem is nearing an equilibrium state.

4 Case Study: Eclipse Open-Source Ecosystems

In this section, we present our case study of eclipse ecosystem [10]. We remark here that the data used in section is extracted from the Eclipse web site, dashboard queries [11].

4.1 Overview of Eclipse

Eclipse is an open-source software development codebase, which contains various software components (plug-ins) that support multiple languages and diverse applications. The owner of Eclipse is the Eclipse Foundation, which is a nonprofit organization. Since Eclipse components are well received, many other organizations use Eclipse to build their own products and services. To make sure that Eclipse components can better meet their business need, these software organizations joined the membership of Eclipse community. Through making suggestions and contributions to Eclipse components, the member organizations could benefit from their desired functionality and quality of Eclipse components. Currently, there are 14 strategic members, 3 enterprise members, 92 solution members, and 69 associate members. In total, 178 organizations have joined Eclipse membership and are contributing to Eclipse components.

Eclipse started as a Java IDE, and has grown to be a collection of open-source components that support static and dynamic languages, client-server development, embedded and mobile computing, modeling, and reporting [10]. Employees of the member organizations can choose to join their interested Eclipse projects and become contributors. To contribute, they need to join as a user or as a committer, where a user can report a bug, post questions, and seek solutions, while a committer can make changes to the components.

For each Eclipse component, there could be contributors from more than one software organization. These software organizations not only contribute to this component, but also depend on this component. Therefore, a software ecosystem is formed around each Eclipse component, which includes the component and all its contributing members (software organizations). Figure 1 shows the ecosystem of Eclipse Platform project, which includes 11 different members.

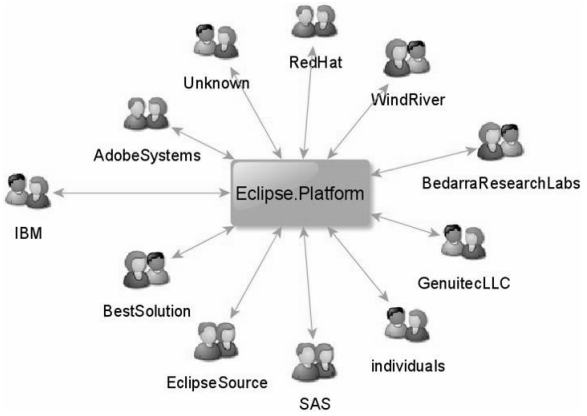

Figure 1: Eclipse platform project ecosystem

4.2 The Evolution of Eclipse Software Ecosystem Entropy

Currently, the contributing organizational data is available from 121 active projects listed in the Eclipse web site. These 121 projects are contributed to by 87 software organizations/companies. To make it easy analyze the data, we assemble similar projects as one project group. For example, projects datatools, datatools.connectivity, datatools.enablement, datatools.modelbase, datatools.incubator, and datatools.sqltools are grouped together as the 'datatools' group in our analysis. Using this scheme, 9 project groups, which contain 117 projects, have been formed. Projects birt, stp, stp.bpmnmodeler, and stp.sca have few data and hence have been omitted from the grouping and are not included in this study.

Table 2 illustrates the details of the 9 project groups, in which CO represents contributing organizations; the total lines changed is measured with the number of thousands of source code line (KLOC) changed in 8 years (2001-2008). Figure 2 shows the number of lines changed in eight years of each project group. Therefore, the total of the bars in project group of Figure 2 matches the corresponding KLOC changed column in Table 2.

The number of line changes represents the aggregated amount of contributions from different software organizations. For example, project group eclipse has 19 contributing members, who have committed over 49M of lines change. As discussed in Section 3, software ecosystem entropy can be used to represent and measure an open-source component's popularity and diversity. We study the evolution of the ecosystem entropy of these nine groups of eclipse components. The

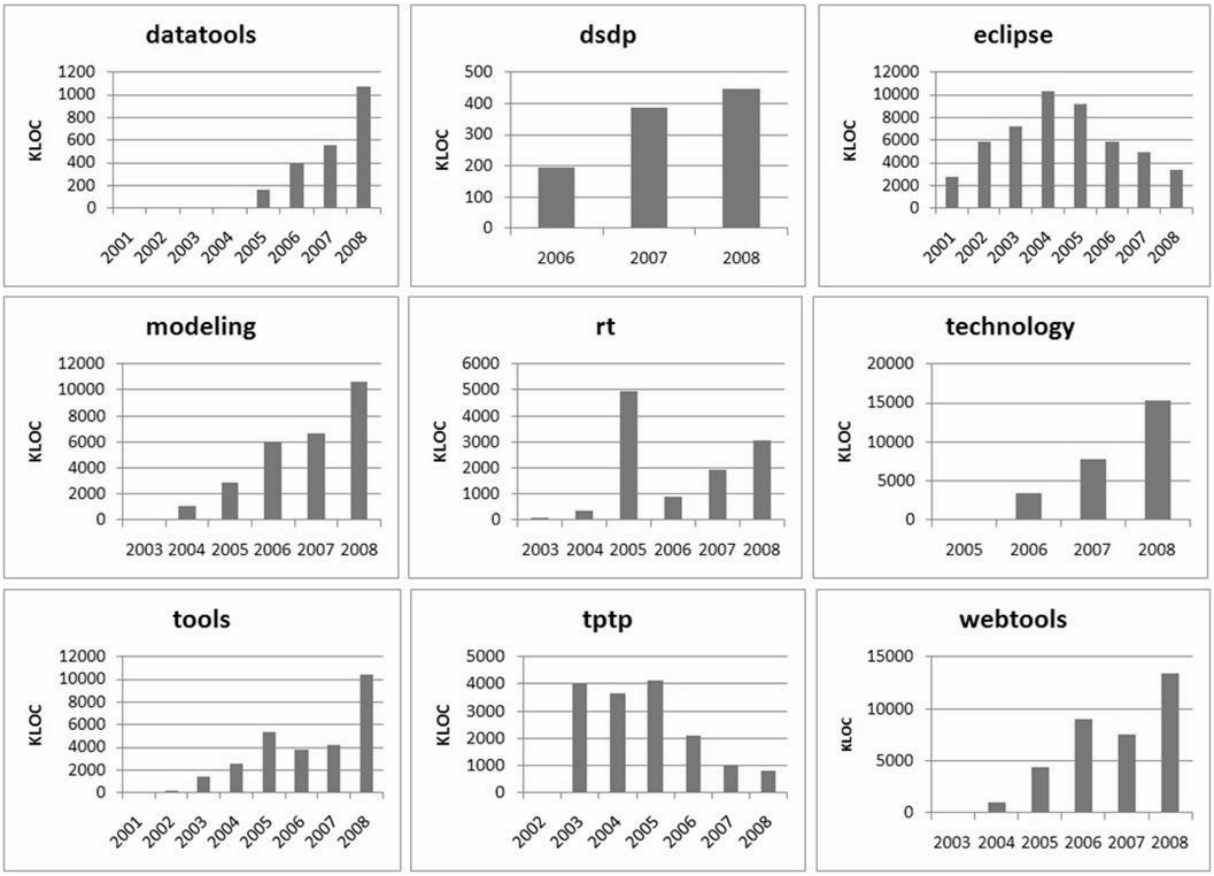

Figure 2: The number of committed line changes of the nine project groups

Table 2: Details of the nine project groups

Project Group	*Numberof Projects*	*Number ofCO*	*KLOC changed*
datatools	6	10	2,196
dsdp	6	11	1,027
eclipse	5	19	49,685
modeling	22	20	27,413
rt	10	19	11,243
technology	38	40	26,548
tools	13	28	28,003
tptp	5	6	15,720
webtools	12	15	35,366

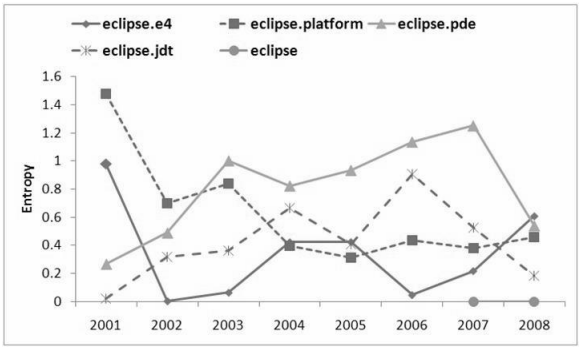

Figure 4: The evolution of the ecosystem entropy of eclipse projects

results are illustrated in Figure 3. It can be seen that for most Eclipse project groups, their ecosystem entropy increases with time, which means these projects are becoming an attractive alternative for commercial software companies.

However, it is interesting to see that the popularity of the representative component-eclipse group, has stabilized in recent years. Examining this collectively with the corresponding number of committed line changes in Figure 2, we can see that the amount of changes increased from 2001 and reached its highest level in 2004, after which it begins to decrease (right about the same time where the entropy also begins to flatten around 0.5). One explanation could be that components in eclipse group have attained marketplace stability around this timeframe (2004). Despite its wider use and adoption after 2004, by different software organization/companies, fewer and fewer organization/companies are making contributions in terms of committing changes. This demonstrates a level of marketplace stability that has been achieved. To further investigate this interesting phenomenon, we studied the evolution of the ecosystem entropy of individual component in eclipse group and the results are illustrated in Figure 4. We can see that in 2001, the popularity of eclipse.e4 (Java IDE) and eclipse.platform (a universal tool platform) was high. In the rest of the years, their popularities are pretty much stabilized.

Another interesting project group is tptp. In Figure 2, it shows that in general, its number of committed line changes demonstrate a decreasing trend from 2001 to 2008. In contrast, its ecosystem entropy in Figure 3 has an increasing trend. This phenomenon can be explained as follows: from 2001 to 2008, tptp components were becoming more stable and fewer changes were being made. However, because of its increasing adoption, more and more software organizations/companies are making contributions (committing changes) to them, although these changes are becoming smaller. Figure 5 presents the detailed ecosystem entropy evolution of every component in project group tptp.

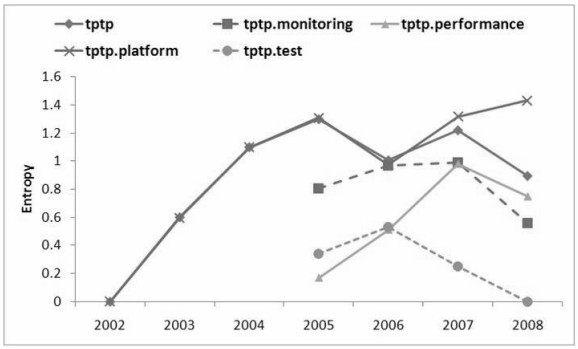

Figure 5: The evolution of the ecosystem entropy of tptp projects

As described before, the value of entropy represents the diversities of contributors. The larger value of entropy indicates higher diversity of the contributors, which also means that the product has a higher popularity. Increasing of product entropy means the increasing of the popularity and market share of the product. From the viewpoint of marketing, we would like to increase the entropy of the product. If the entropy value becomes stable, it could means that the product has reached its highest possible popularity within current capacity.

5 Threats to Validity and Future Research Directions

There are some significant threats to the validity of this study. They are listed below.

We have used entropy to represent a software components' increasing popularity among companies and the diversity of its use. Our measurement of entropy can be termed 'contribution entropy' , i.e. one which measures the diversity of contributors to the open source efforts. Studying contributors' diversity may be driven by special interests and market pressures; and we have

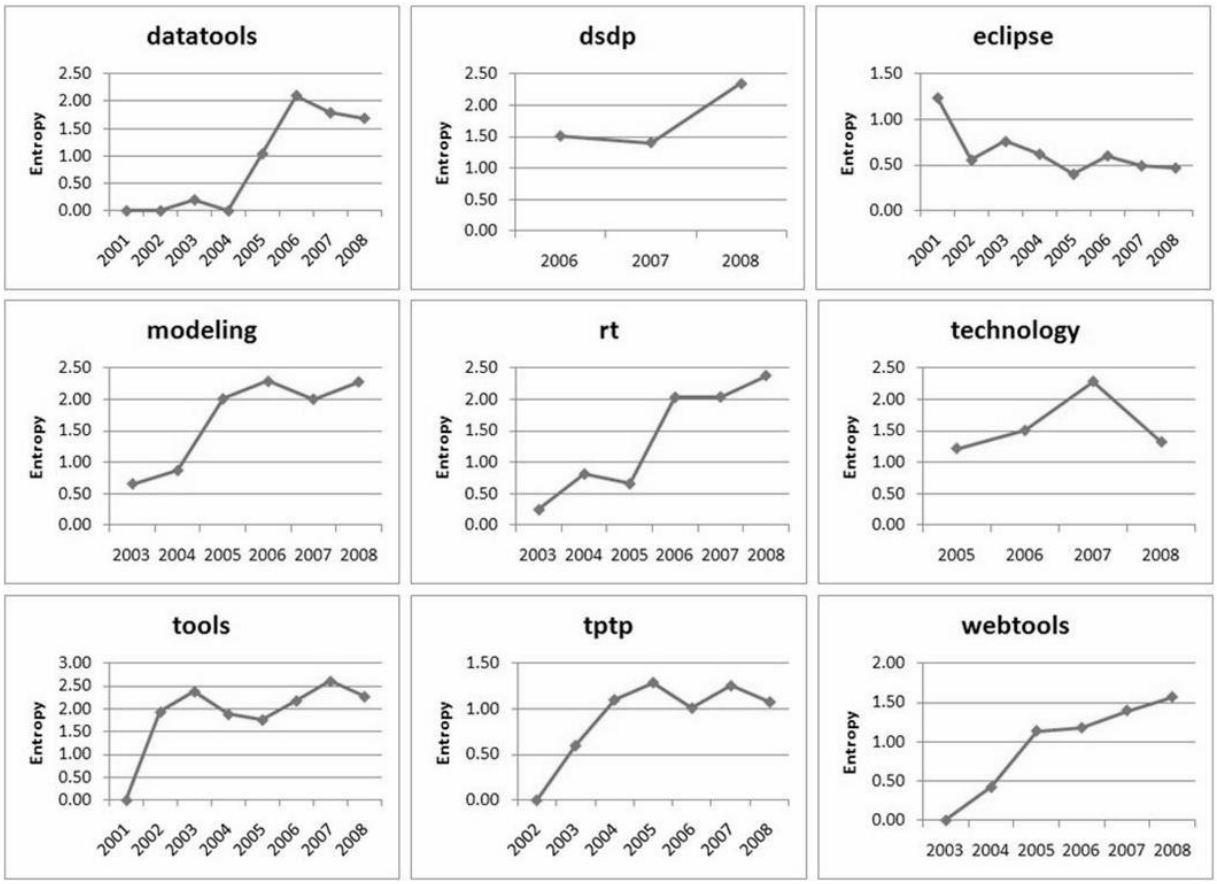

Figure 3: The evolution of the ecosystem entropy of nine Eclipse project groups

assumed it to be a leading indicator of user popularity. While this is potentially true for Eclipse, this may not be always true for other software systems. Further research should be carried out to find other related measurements of entropy and its application to studying diversity in complex software systems development and use.

The variability in the entropy can be further explored vis-à-vis its relationship to stability. For example: Our study finds that within the Eclipse community, when viewed as an Eclipse ecosystem, most component entropies have increased during the past several years. In some components, such as their representative eclipse group (Java IDE), the entropy indicates a degree of marketplace stability that has been achieved.

Defining a normalization measure that is independent of when companies choose to join these groups may be an interesting twist to understand the vested roles of these participants better. This paper presents entropy as a measure that could attract further research focus. However, for unifying and generalizing these related concepts of popularity, stability and entropy better, further research is needed, possibly using another open-source software ecosystem - for example, Ubuntu.

6 Conclusions

In this exploratory paper, we adapted the concept of entropy from thermodynamics and information theory and applied it for studying the marketplace stability of software components in an open-source ecosystem.

Software systems have become an integral part of our everyday lives and often multiple software components are integrated into complex industrial software systems. By virtue of such integration, such components often have critical and significant implications to our quality of life issues. Hence studying the evolution of these systems as an interconnected ecosystem becomes an important and challenging issue. Applying an interdisciplinary approach to studying such complex software systems thus becomes a dire necessity for developing a holistic understanding of the various components that make up such software system and their evolution. Through this study, we have demonstrated that the entropy measure could be used as a tool to study software component stability by examining corresponding factors such as component popularity and user diversity.

References

[1] Alvim, M., Andrés, M., and Palamidessi, C. Entropy and attack models in information flow. *Theoretical Computer Science: IFIP Advances in Information and Communication Technology*, 323:53–54, 2010.

[2] Berra, D. L. and Rapicault, P. Dependency management for the eclipse ecosystem: eclipse p2, metadata and resolution. In *Proceedings of the 1st International Workshop on Open Component Ecosystems*, pages 21–30, Amsterdam, The Netherlands, 2009.

[3] Bianchi, A., Caivano, D., Lanubile, F., and Visaggio, G. Evaluating software degradation through entropy. In *Proceedings of the 7th International Symposium on Software Metrics*, 2001.

[4] Bosch, J. From software product lines to software ecosystems. In *Proceedings of the 13th Software Product Line Conference*, San Francisco, CA, USA, 2009.

[5] Bosch, J. and Bosch, P. From integration to composition: on the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67–76, 2010.

[6] Bosch, J. and Bosch, P. Software product lines, global development and ecosystems: collaboration in software engineering. In *Collaborative Software Engineering*. Springer, 2010.

[7] Boucharas, V., Jansen, S., and Brinkkemper, S. Formalizing software ecosystem modeling. In *Proceedings of the 1st International Workshop on Open Component Ecosystems*, pages 41–50, Amsterdam, The Netherland, 2009.

[8] Casebolt, J., Krein, J., MacLean, A., Knutson, C., and Delorey, D. Author entropy vs. file size in the gnome suite of applications. In *Proceedings of 6th IEEE Intnational Working Conference on Mining Software Repositories*, pages 91–94, Vancouver, Canada, 2009.

[9] Chen, L., Li, L., Hu, Y., and Lian, K. Information security solution decision-making based on entropy weight and gray situation decision. In *Proceedings of the 5th International Conference on Information Assurance and Security*, pages 7–10, Xian, China, 2009.

[10] Eclipse.org. http://www.eclipse.org/, last accessed on March 16, 2012.

[11] Eclipse.org. Eclipse Dashboard Queries, http://dash.eclipse.org/dash/commits/web-app/, last accessed on March 16, 2012.

[12] Gopalakrishnan, R. The entropy of markets. *Business Line*, 2007.

[13] Hanssen, G., Yamashita, A., Conradi, R., and Moonen, L. Software entropy in agile product evolution. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, 2010.

[14] Hassan, A. Predicting faults using the complexity of code changes. In *Proceedings of the 31st Intnl. Conf. on Software Engineering*, pages 78–88, Vancouver, Canada, 2009.

[15] Hassan, A. and Holt, R. C. The chaos of software development. In *Proceedings of the 6th International Workshop on Principles of Software Evolution*, pages 84–94, Helsinki, Finland, 2003.

[16] Hunt, A. and Thomas, D. *The Pragmatic Programmer*. Addison Wesley, 1999.

[17] Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press, Addison-Wesley, 1992.

[18] Jos, J., Trienekens, R. J., Kusters, D., and Kriek, P. Entropy based software processes improvement. *Software Quality Journal*, 17(3):231–243, 2009.

[19] Jung, W., Lee, E., Kim, K., and Wu, C. A complexity metric for web applications based on the entropy theory. In *Proceedings of the 15th Asia-Pacific Software Engineering Conference*, pages 511–518, Beijing, China, 2008.

[20] Krein, J., MacLean, A., Delorey, D., Knutson, C., and Eggett, D. Language entropy: a metric for characterization of author programming language distribution. In *Proceedings of the 4th Workshop on Public Data about Software Dvelopment*, pages 6–12, Skovde, Sweden, 2009.

[21] Lehman, M. and Belady, L. *Program evolution: processes of software change*. Academic Press Professional, Inc., San Diego, CA, USA, 1985.

[22] Maxwell, J. *Theory of Heat*. Dover, 2001.

[23] McCauley, J. Thermodynamic analogies in economics and finance: instability of markets. *Physica A*, 329:199–212, 2003.

[24] Messerschmitt, D. G. and Szyperski, C. *Software Ecosystem: Understanding an Indispensable Technology and Industry*. MIT Press, Cambridge, MA, USA, 2003.

[25] Olague, H., Etzkorn, L., and Cox, G. An entropy-based approach to assessing object-oriented software maintainability and degradation - a method and case study. In *Proceedings of International Conference on Software Engineering Research and Practice*, pages 442–452, Las Vegas, Nevada, USA, 2006.

[26] Rahmani, H., Sahli, N., and Kammoun, F. Joint entropy analysis model for ddos attack detection. In *Proceedings of the 5th International Conference on Information Assurance and Security*, pages 267–271, Xian, China, 2009.

[27] Sandroni, A. Market selection when markets are incomplete. *Journal of Mathematical Economics*, 41(1-2):91–104, 2005.

[28] Selvarani, R., Nair, T., Ramachandran, M., and Prasad, K. Software metrics evaluation based on entropy. In *Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization*. IGI Global, 2010.

[29] Shannon, C. A mathematical theory of communication. *The Bell System Technical Journal*, pages 379–423, 1948.

[30] Song, H., Wu, D., Li, M., Cai, C., and Li, j. An entropy based approach for software risk assessment: A perspective of trustworthiness enhancement. In *Proceedings of the 2nd International Conference on Software Engineering and Data Mining*, pages 575–578, Chengdu, China, 2010.

[31] Taylor, Q., Stevenson, J., Delorey, D., and Knutson, C. D. Author entropy: a metric for characterization of software authorship patterns. In *Proceedings of the 3rd International Workshop on Public Data about Software Development*, Milan, Italy, 2008.

[32] Yin, B., Zhu, L., and Cai, K. Entropy-based measures of heterogeneity of software structural profile. In *Proceedings of the 34th IEEE Annual Computer Software and Applications Conference Workshops*, pages 196–201, Chengdu, China, 2010.

[33] Yu, L., Ramaswamy, S., and Bush, J. Software evolvability: an ecosystem point of view. In *Proceedings of the 3rd International IEEE Workshop on Software Evolvability*, pages 75–80, Paris, France, 2007.

[34] Yu, L., Ramaswamy, S., and Bush, J. Symbiosis and software evolvability. *IT Professionals*, 10(4):56–62, 2008.

[35] Yu, L., Ramaswamy, S., and Lenin, R. B. Entropy studies of software market evolution. In *Proceedings of the 8th Annual Conference on Applied Research in Information Technology*, pages 87–91, University of Central Arkansas, Conway, Arkansas, 2009.