Software Engineering Ecosystems

Nykolay Sydorov

National Technical University of Ukraine Igor Sikorsky Kyiv Polytechnic Institute, street Politechnichna, 41. Kyiv, 02000, Ukraine

Abstract

Nowadays, the fundamental science of software engineering is being formed, which should represent knowledge that meets the requirements of the concept of sustainable development. This the fundamental science can be called the Software Engineering Ecology. Along with others sections, the Software Engineering Ecology should include a section containing knowledge about software engineering ecosystems. This section of the future science has been intensively developing for more than fifteen years, exploring software ecosystem. However, today, there is no consensus among researchers regarding the definitions of the software ecosystem. Naturally, this does not contribute to the creation of an appropriate section, an emerging science. Being investigated only a software ecosystem, which is considering in different contexts and defining in different ways. Based on the hypothesis that the term "the software ecosystem" is now used to refer to a wide range of ecosystems that are actually software engineering ecosystems, the purpose of this paper was to propose a basis for defining software engineering ecosystems. As such a base, by analogy with the concepts of the landscape and the trophic chain of biological ecosystems, the concepts of software landscape and software engineering value chain are proposed. Based on these concepts, the diversity of software engineering ecosystems is shown. A model of the software engineering ecosystems and a classification of the software engineering ecosystems are proposed.

Keywords

Software engineering, software ecosystem, landscape, value chain, ecosystem model, ecosystems types, software engineering ecosystem

1. Introduction

Software being the result of solution to software engineering problem is always a product. It has a user and an operating environment. The product is created and transferred to the customer or buyer in the context of the life cycle and must meet a number of requirements specific to products of any engineering (product design, quality, standards, documentation, economics, maintenance, environmental impact in the context of sustainable development). The life cycle is a system-forming factor in software engineering, as it defines the processes, resources and products used and created by engineering. This feature of software engineering also determines the structure of knowledge inherent in it, which can be represented as a layered cylinder. The horizontal layers of the cylinder represent the fundamental sciences of software engineering, and the vertical division corresponds to the knowledge regarding the practical implementation of the phases of the software life cycle. It is obvious that the fundamental sciences are applied in any vertical section of the cylinder. For example, Software engineering economics, Software engineering culture, Computing Foundations are the fundamental sciences [1]. The fundamental sciences should also include the now emerging science, which, by being included in the vertical sciences, should supply knowledge that meets the requirements of the concept of sustainable development. This may be the Software Engineering Ecology [2, 3]. Along with others sections, the Software Engineering Ecology should include a section containing knowledge about software engineering ecosystems. This section of the future science has been intensively developing for more than fifteen years, exploring software ecosystem. However, today, there is no consensus among researchers regarding the definitions of the software ecosystem. Naturally, this does not contribute to the creation of an appropriate section, an emerging science. Being investigated only a software

13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: NykSydorov@gmail.com
ORCID: 0000-0002-3794-780X

ecosystem, which is considering in different contexts and defining in different ways. Based on the hypothesis that the term “the software ecosystem” is now used to refer to a wide range of ecosystems that are actually software engineering ecosystems, the purpose of this paper was to propose a basis for defining software engineering ecosystems. As such a base, by analogy with the concepts of the landscape and the trophic chain of biological ecosystems, the concepts of software landscape and software engineering value chain are proposed. Based on these concepts, the diversity of software engineering ecosystems is shown. A model of the software engineering ecosystems and a classification of the software engineering ecosystems are proposed.

2. Toward the software engineering ecosystems definition

To define the software engineering ecosystem, this article introduces two concepts similar to the basic concepts of biology. It is a landscape and value chain that matches the landscape and food chain of natural ecosystems. The totality of landscapes of all activities of the value chain of the software engineering forms the software engineering territory. Taking into account the diversity of biotopes in the value chain, we can talk about the diversity of software engineering ecosystems.

2.1. The software engineering ecosystem landscape

According to the definition of ecology, an ecosystem (biogeocenosis) is defined as a supraorganismal system of interacting biotic (living) and abiotic (non-living) components located in a certain territory [4, 5]. In ecology, when an ecosystem is defined, always a certain area, or space, terrain, landscape is defined. For example, the ecosystem of a forest, lake, pond. It is an important component of the ecosystem. Therefore, when defining the software engineering ecosystem, it is expedient to define a similar concept.

Four candidates can be used as a metaphor for the similar concept of the software engineering ecosystem. These are territory, environment, terrain and landscape. The term "territory" usually refers to a piece of space that can be clearly defined. Usually, this is the territory of the country. This term is proposed to be applied at the level of the software engineering. The term "environment" has long and consistently been used for the software development environment [6]. The term "terrain" refers to a specific space that is already landscaped with the biotic component of the ecosystem. In addition, it is used in geographic information systems when visualizing space. This term can be analogous to biotope [5]. Finally, the term "landscape" is used to refer to a space, whether or not it is landscaped. In the software engineering, this term is used as a metaphor for the visualization of a software system [7]. In our opinion, it is this term that is most suitable for designating the software engineering ecosystem space, taking into account its initial independence from the activity of the biotic component.

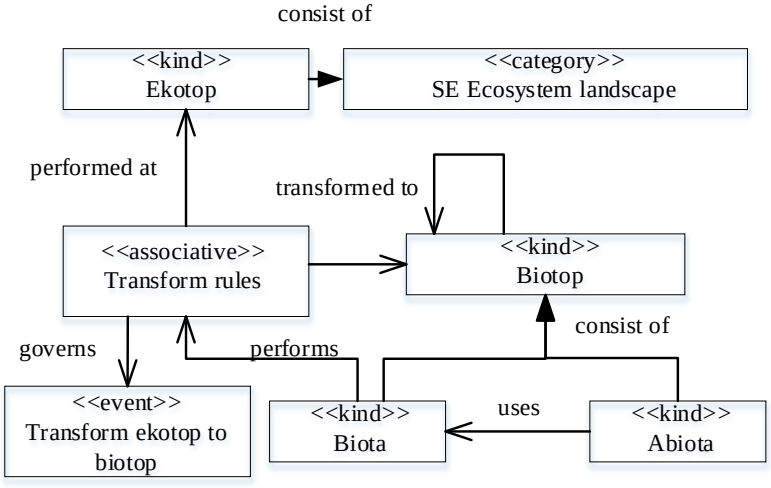

Figure 1: Transform the landscape (ecotope) into a biotope

Thus, initially, the landscape, being an ecotope, as a result of transformations (activities) carried out by the biota is transformed into a biotope, turning, for example, into a terrain (Figure 1). In this context, this transformation is important for two reasons. First, unlike ecology, the landscape of an ecosystem in software engineering cannot initially be specified (it is not a forest, a field, a pond, etc.). However, this can be done if the landscape and the activity of the biota to join. Secondly, the activity of the biota will determine the nature of the biotope in the future. Therefore, biotopes will be different for different ecosystems even with the same activity. This is explained by the fact that the rules of transformation and, consequently, the result of the transformation of an ecotope into a biotope for biotas of the same activity can be different.

2.2. The software engineering value chain

The ecosystem of biology is defined as the unity of interacting biotic (living) and abiotic (non-living) components, which, based on the energy flow, establishes the trophic structure of the ecosystem, species diversity and the circulation of substances in it [4]. Therefore, the second important part of an ecosystem is its trophic structure. In defining the software engineering ecosystem, we will follow this definition. However, instead of the trophic structure, we will use the added value factor of the same importance for software engineering. Then, instead of the trophic structure in software engineering, we will use the value chain of value-based software engineering [8].

To build a chain, we will use nodes of two types (Figure 2, a, b). The first type of node designates the activity that corresponds to the process of the chain (Figure 2, a), indicating in it the designation of the process (Activity) and indirectly the landscape, the designation of the biotic component (Biota), which implements the process and the abiotic component (Abiota), which the biotic component uses to implement the process. The second type of node, we use to denote the added value - AVi (Figure 2, b).

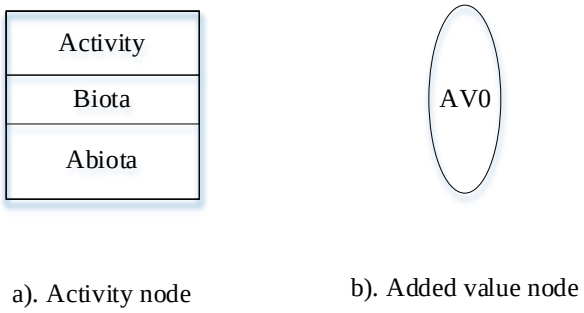

Figure 2: Types of value chain nodes

The value chain for software engineering can look like this (Figure 3).

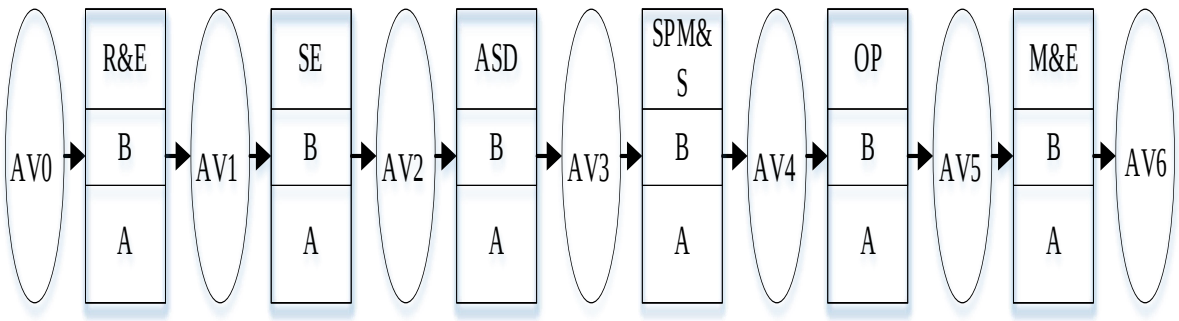

Figure 3: Software engineering value chain

On the Figure 2, the following designations are used:

- AV0 - added value from previous activities (in this case, these are the results of research in fundamental sciences that are not related to software engineering);

- R&E - Research and Education activity - fundamental research and education in software engineering; the main biotic component consists of researchers and teachers, as well as other actors who perform supporting roles, such as trainers, publishers, editors, etc.; abiotic component consists of educational standards and software, accreditations, universities, journals, conferences, professional organizations, for example, ACM, IEEE;

- AV1 - added value from R&E activities, for example, theories, approaches, principles aimed at the development of software engineering, and engineers, scientists prepared to work in software engineering;

- SE - Software Engineering activity - applied research in software engineering; the main biotic component, these are software engineers, as well as other actors who perform supporting roles, for example, technical and maintenance personnel; the abiotic component consists of professional websites, magazines, organizations such as Association of Software Professionals , Association for Women in Computing , Python Software Foundation , IACSIT Software Engineering Society , GitHub;

- AV2 - added value from SE activities, such as platforms, tools, APIs, reusable components for use in the processes of creating and maintaining application software;

- ASD - Application Software Development activity - life cycle processes focused on the creation of application software; the main biotic component consists of software developers, as well as other actors who perform supporting roles, for example, the actors of supporting processes, products and tools for creating application software; abiotic component, - resources and materials necessary for the implementation of life cycle processes, standards, documentation;

- AV3 - added value from ASD activity, - software product for the application domain;

- SPM&S - Software Product Marketing and Sale activity - processes aimed at the software product market and sales; the main biotic component consists of product managers, market analysts and sales managers; abiotic component consists of resources and materials necessary for the delivery of the software product to users;

- AV4 - added value from SPM&S activity - software product sold or delivered to the user;

- Op - Operation activity - use of the software product; the main biotic component consists of users, as a rule, from the application domain and the software product maintenance group, as well as other actors ; abiotic component consists of resources and materials necessary for the use of the software product;

- AV5 - added value from Op activity, - consultations, user trainings, changes that have been made to the product and documentation;

- M&E - Maintenance and Evolution activity - maintenance and evolution of the software product; the main biotic component consists of maintainers, domain analysts, workers for reuse and reworking and other actors; abiotic component consists of supported software product, technical resources for maintenance and evolution;

-AV6 - added value from M&E activity, - an evolving software product, parts of a legacy software product that are sent to the reuse as components to create and maintain a new software product.

2.3. The software engineering ecosystems territory

By analogy with the ecosystems of ecology, the landscape metaphor is proposed to use for software engineering ecosystems. An ecosystem is a landscape on which the unity of interacting biotic (living) and abiotic (non-living) components is determined to perform the processes of the corresponding activity. Landscapes of software engineering ecosystems can be distinguished by the unity of these components. Unity is determined by focusing on the processes aimed at obtaining the corresponding added value. Landscapes can be local or distributed geographically. Local landscapes are homogeneous environments, whit the same technical, social and cultural values, that correspond to the geography of the given landscape. A distributed landscape, if it occupies geographically different territories, obviously cannot be characterized in this way. The totality of landscapes of all activities of the value chain of the software engineering forms the software engineering territory.

2.4. Diversity of ecosystems

Taking into account the diversity of biotopes in the value chain, we can talk about the diversity of software engineering ecosystems. The landscape of each ecotope within a value chain node can be thought of as being composed of other landscapes and thus defining ecosystems corresponding to the nested landscapes. In ecology, this corresponds to the concept of an elementary landscapes [4]. The application of this view can be illustrated by the example of ASD activity, taking into account the life cycle processes that are used to create application software. Considering, for example, the sequential software life cycle model, it can be represented as a value chain, where each activity will unfold on the corresponding landscape (Figure 4). Landscape together with biota and abiota will represent the ecosystem.

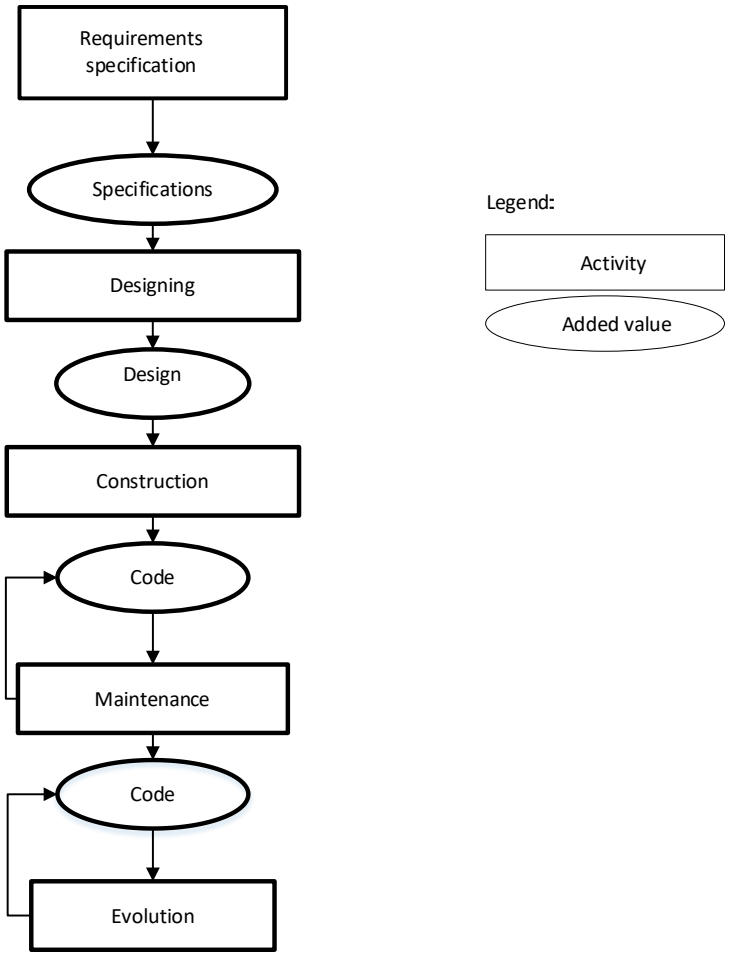

Figure 4: The sequential software life cycle model as the software engineering value chain

Similarly, incremental, evolutionary, spiral and other software life cycle models based on the sequential life cycle model can be considered. The same view can be applied to models of Agile methodology.

Finally, the diversity of software engineering ecosystems can be extended to the engineering that are parts of it. This applies to the reverse software engineering and the empirical software engineering. For example, for reverse software engineering, can build a value chain, where each added value will correspond to the knowledge obtained as a result of software analysis activity at the corresponding level of its presentation (requirements specifications, design, and source code). United of landscape, biota, and abiota each of the corresponding level of presentation can be ecosystem.

A diversity of activities and biotopes will create a diversity of software engineering ecosystems. We propose to define the software engineering ecosystem using the term "landscape" and the designation of

the corresponding activity in the value chain. For example, for the activity «Application Software Development» is defined "the Ecosystem of Application Software Development Landscape".

3. What about the software ecosystem?

From the point of view of this work, the factor that plays a fundamental role in the definition of an ecosystem is the landscape. It, in the sense taken here earlier, is absent in the known definitions of the software ecosystem [13]. However, the term "software ecosystem" can be used if software is a landscape. Obviously, then we can talk about some a software system, the parts of which, being in relationships and interacting, will represent the abiotic component of the ecosystem. The spatial structure of this abiotic component will be represented by the software landscape, which can be visualized [7]. At the same time, obviously, the software landscape will be is a terrain that is a biotope, which, strictly speaking, does not contain biota. However, it is also possible to represent the biotic component of the software landscape, if, for example we used the parallelism between the abiota components of the Software ecology and the biota components of Environmental Biology (Table). Then, for example, the abiotic component can be data in different forms.

Table

Software Ecology and Environmental Biology Analogies

№	Software Ecology	Environmental Biology
1.	Alphabet	Molecules
2.	Lexemes, simple types	Cells
3.	Operators, complex types	Tissues
4.	Structured operators	Organs
5.	Subroutines	Organs (organisms)
6.	Modules (Classes, Objects)	Organs (organisms)
7.	Megamodules	Organisms
8.	Software life cycle products	Population
9.	Software products line (family)	Community
10.	Software of system of systems	Ecosystem
11.	Information flows	Food chains
12.	Cycles of reuse	Nutrient cycling

By analogy with ecology [4], for software ecosystem may take place tasks of studying the space-time structure of the software ecosystem, information and control flows (algorithms), the principles of software evolution and reusable software components circulation (Subroutines, Modules (Classes), Megamodules), when legacy software is reused [3].

4. Software engineering ecosystem model

On the Figure 5 presents a model of the software engineering ecosystem. Initially, the space that the ecosystem will occupy is the landscape. Then, a biota is located on the landscape. After that, the landscape can be considered as an ecotope. The biota begins the activity of transforming the ecotope. The ecotope is transformed into a biotope. It is important to determine the nature and results of the biota's activity in transforming an ecotope into a biotope. This can be done using of the software engineering culture [9]. The main difference between a biotope and an ecotope is the best conditions for performing an activity. The change of the ecotope conditions is aimed at getting more added value after performing this activity. The means to implement change will be different maturity models (CMMI, P-CMM), standards, culture, ethics code, organizational paradigms, and platforms. For example, in [10] for the change of the ecotope conditions for creating the environment, tools, and activities of DevSecOps

ecosystem the preparation phase is used. The phase includes understanding the amount of cultural and process change that is required, and identify resources and a strategy that will be transformed the current culture into one that supports DevSecOps principles and behaviors.

For example, aspects transforming of an ecotope into a biotope can include the following:

- ensuring that biota has or will have the skills needed to fulfill of project;
- ensuring that biota has or will have the infrastructure and the resources (abiota) needed to fulfill of the project;
- ensuring that the software processes will be enough mature (desirable accordance CMMI, P-CMM);
- ensuring the developers, testers, security officers, and maintainers which will be the team understand their roles and expectations;
- ensuring the culture of organization, honesty and transparency, and group dynamics and communications norms are understood and followed;
- preparing upstream and downstream stakeholders for the workflow (and the associated changes in process), and setting expectations on performing their responsibilities [10];

The current state of biotope is not be static forever. Inevitably, will need modifications. The biotope will evolve over time, and better options will arise and be adopted. At the same time, in the future, in order to increase the added value, changes can be carried out in parallel with the performing of biota the main activity using the abiotic component of the biotope.

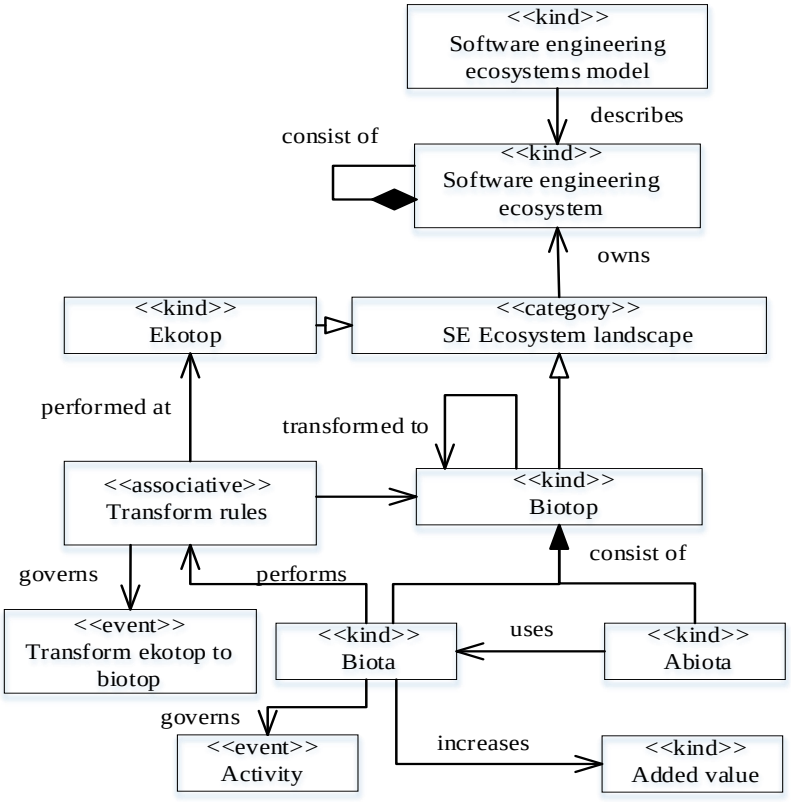

Figure 5: Software engineering ecosystem model (SE - Software Engineering)

5. Types of the software engineering ecosystems

The Figure 6 shows the classification of the software engineering ecosystems. According to the nature of the landscape, all ecosystems can be divided into two types - ecosystems of the human activity landscape and ecosystems of software landscape.

Ecosystems of the first type initially occupy some undeveloped space - an ecotope, which is being developed and transformed into a biotope. Further, ecosystems of the first type is divided into two

groups. In the first group, we include ecosystems of engineering included in the software engineering, and in the second group, ecosystems of non-engineering landscapes (R&E, SE, SPM&S) (Figure 3).

Ecosystems of the second type is represented by an equipped landscape - software landscape (terrain, biotope). According to the type of software landscape, ecosystems of the second type can be divided into three groups. Software life cycle products (software systems) as ecosystems form the first group. In this case, the software ecosystem should be considered as a population of organisms and, therefore, as a system of the supraorganismal level (Table). The second group is formed, if it is take into account the concept of the Individual-based Modeling [11]. Ecosystems are based on individuals and the properties of individuals determine the properties and behavior of the ecosystem as a whole. Therefore, it makes sense to model individuals (in this case the individuals are a software artefacts) in the context of a software ecosystem. They are named "the individual-based software ecosystem". For example, the programing style as the software artifact of the individual-based software ecosystem [12]. The third group is formed by software ecosystem of the system of systems ecosystem, in which software products of the various owners act as Populations (Table). For example, big data software ecosystem consists of the software products for data collection (Data ingestion, Data loading and preprocessing), reparation (Information extraction, Data cleaning, big data integration), analytics (Data analysis, Data loading and transformation), and visualization (Data visualization).

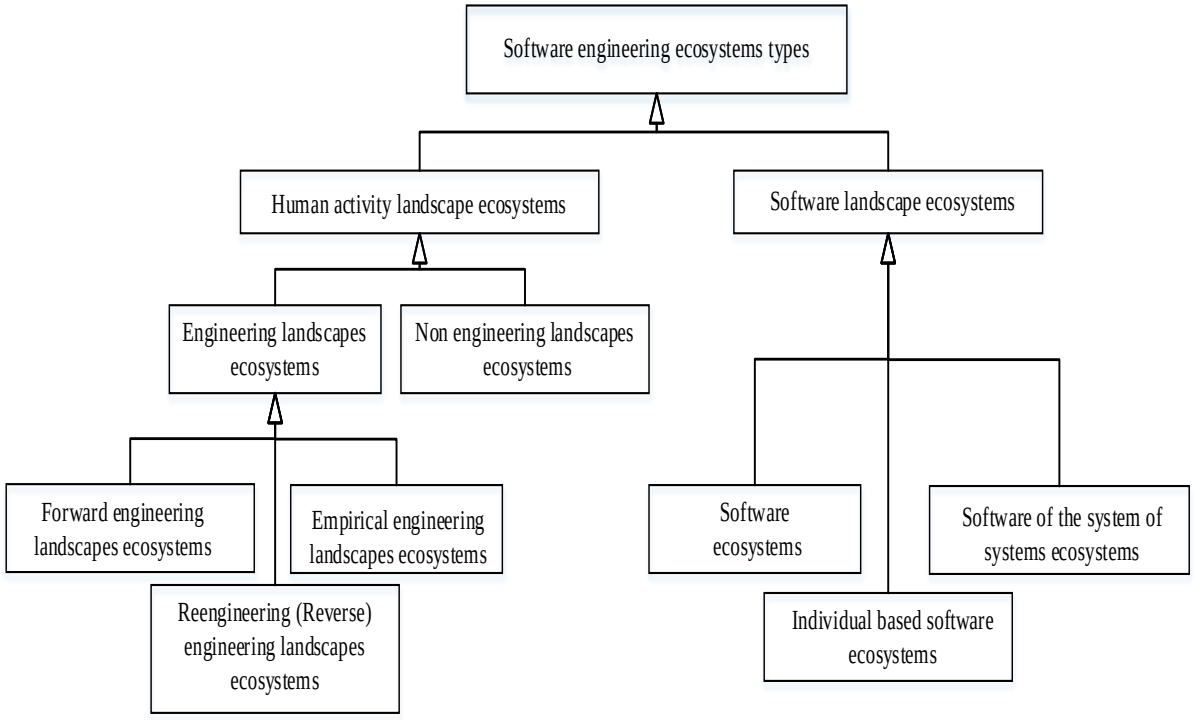

Figure 6: The types of the software engineering ecosystems

Can assume, that several or all types of ecosystems can take place in the ecosystems of large companies, for example, IBM, SAP, Microsoft [13].

6. Related works

The literature on software ecosystems is extensive and presents ecosystem definitions, requirements, models, and case studies. The state of research can to know using systematic reviews, for example, works [13 - 16].

In the work [13] stated that out of 90 analyzed works, 40 do not define the software ecosystem, but use definitions from other works. In the remaining 50 papers, four definitions are used with references

to the authors (in [15], six definitions are indicated). However, these four definitions also so different, that the authors of work [13] had to look the common properties in order to formulate one definition. The common properties are the followings [13]: Common Software, Business, and Connecting Relationships. Combining the definitions with the three properties identified, a software ecosystem was defined as the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. Obviously, this suggests that among researchers there is no consensus on the definition of the term "software ecosystem". The ambiguous state of affairs is clearly seen from the results of the work [16], which sets the goal of is to develop meta-model that should help researchers to describe and structure the software ecosystems they are investigating. Examining the literature on five topics that relate to the components of software ecosystems, the authors present the following results. There are 46 types of entities found on the topic "actors and roles" (in the work [15] identified over 90 roles). At the same time, there is a huge spread of actors in the list, from a Researcher to a Bank and an Investor. On the topic "the Products" and "the Platforms", the same state is observed. A total, 27 entities were found, ranging from API to Use case. On "the Strategy" topic, 21 entities have been identified, and the spread is still large, from the Product lifecycle strategy to Licensing. Finally, on "the Boundaries" topic, seven entities are identified, and the range is from Abstraction level to Output. In the work [16], only eight papers were identified as using the definition of the software ecosystem boundary. The term "the Environment" is used in three works as a context in which software products or services operate, or a context in which a collection of software projects are developed and coevolve together [17 – 19]. Analyzing the composition of the entities indicated in [16], one should pay attention not so much to their number and diversity, but most importantly to their disunity. For example, the finding of the "Researcher" and "Hedger" in "the actors" topic or "Community driven" and "Executable components" (in "Products and Platforms" topic) in the same software ecosystem it is like the finding of the actors "the Lion" and "the Penguin" in one biological ecosystem. It can be said about other entities of the meta-model. Obviously, this indicates that the ecosystem or its boundaries are incorrectly defined.

In the result of the analysis of related works, it has been suggested that the term software ecosystems is now actually used to refer to a wider range of ecosystems, which are actually software engineering ecosystems. Therefore, the purpose of our work was to propose such a definition of software engineering ecosystems that would allow us to avoid the existing ambiguity.

7. Conclusion

Based on the hypothesis that the term “software ecosystems” is used to refer to a wider range of ecosystems, which are actually software engineering ecosystems, the goal of paper was to proposed a base for defining software engineering ecosystems. As a base, by analogy with biological ecosystems, the concepts of landscape and the value chains of the software engineering are proposed used. Based on these concepts, the existing variety of software engineering ecosystems is pointed out, a model of the software engineering ecosystem and the classification of the software engineering ecosystems are proposed. This paper is a continuation of the author's works on the topic [3, 12, 20 - 22].

In further work, to confirm the thesis of the article, it is undoubtedly necessary to conduct a Case study, using the example of such companies as Microsoft, IBM, Google, SAP and the like, presumably having all types of software engineering ecosystems on their territory.

8. References

[1] P. Bourque, R.E. Fairley (Eds.), Guide to the Software Engineering Body of Knowledge, ver.3.0,IEEE CS, (2014). URL: http://www.swebok.org.

[2] T. N. Nguyen, The Ecology of Software: A Framework for the Investigation of Business-IT Integration Issues and Trends of Information Technology Management in Contemporary Organizations, in: Proceedings of the Information Resources Management Association International Conference, 2002.

[3] N.A. Sydorov, Software ecology, Software engineering, N. 1, (2010) pp. 53-61 (in Ukrainian).
[4] E P. Odum, Fundamentals of Ecology, Saunders Company, Philadelphia-London, 1971.
[5] V.N. Sukachev (Ed.), Fundamentals of forest biogeocenosis, M. Science, 1964 (in Russian).
[6] D. E. Perry, G. E Kaiser, Models of Software Development Environments, IEEE Transactions on Software Engineering, (1991), volume 17, no. 3, 283-295.
[7] M. Balzer, A. Noack, O. Deussen, Software Landscapes: Visualizing the Structure of Large Software Systems, in: Proceedings of IEEE TCVG Symposium on Visualization (VisSym), May 19-21, Konstanz, 2004.
[8] S. Biffl, A. Aurum, B. Boehm, Value-Based Software Engineering, Springer, 2006, 398 p.
[9] K. Wiegers, Creating a Software Engineering Culture, Dorset House Publishing, 1996.
[10] Guide to Implementing DevSecOps for a System of Systems in Highly Regulated Environments, Technical report CMU/SEI-2020-TR-002, April, (2020) 111 p.
[11] V. S. Grimm, F. Railsback, Individual-based Modeling and Ecology, Princeton University Press, 1999, 429 p.
[12] N. Sydorov, Programming Style as an Artefact of a Software Artefacts Ecosystem, in: Proceedings of Advances in Computer Science for Engineering and Education Applications, Springer, Cham, 2021, pp. 244-255.
[13] K. Manikas, K. Hansen, Software ecosystems-a systematic literature review, J. Syst. Software. 86(5), (2013) 1294–1306.
[14] A. García-Holgado, F. J. García-Peñalvo, Mapping the systematic literature studies about software ecosystems, in: Proceedings of TEEM'18. Sixth International Conference on Technological Ecosystems for Enhancing Multiculturalism (Salamanca, Spain, October 24th-26th, 2018), New York, NY, USA: ACM. pp. 910-918. doi: 10.1145/3284179.3284330.
[15] S. Suortti, The Role of Software Platform and Actors in Software Ecosystems: A Case Study in Agriculture, Aalto University, 2017.
[16] J. Wouters, J. R. Ritmeester, A. W. Carlsen, A SECO Meta-model A Common Vocabulary of the SECO Research Domain, in: Proceedings of 10th International Conference, ICSOB 2019 Software Business, Jyväskylä, Finland, November 18–20, 2019,Proceedings Springer Nature Switzerland AG, 2019.
[17] V. Boucharas, S. Jansen, S. Brinkkemper, Formalizing software ecosystem modeling, in: Proceedings of the 1st International Workshop on Open Component Ecosystems, IWOCE 2009, pp. 41–50.
[18] M. F Lungu, Reverse Engineering Software Ecosystems (Philosophy), Ph.D. thesis, submitted to the Faculty of Informatics of the University of Lugano, September 2009.
[19] J. Knodel, K. Manikas, Towards a Typification of Software Ecosystems, Software Business, in: Proceedings of 6th International Conference, ICSOB 2015 Braga, Portugal, June 10–12, 2015, pp. 77-82.
[20] M. Lutcsky, N. Sydorov, The software – an ecological approach to study, in: Proceedings of Natural and Artificial Intelligence Conference, ITHEA, 2010, Sofia, Bulgaria, 2010, pp. 181-189.
[21] N.A. Sydorov, N. N. Sydorova, E.N. Sydorov, Description model of programming style ecosystem, Problems in programming, no. 2-3, (2020), 74-81.
[22] N.A. Sydorov, Toward a software artifacts ecosystem, Problems in programming, no. 4, (2020), 110-120. doi.org/10/15407/pp2022.03-04.092.