ONE APPROACH TO STUDY SOFTWARE ECOSYSTEM PROPERTIES

Olena Grinenko, Sergiy Grinenko

Abstract*: One of the factors that determine the effectiveness of software ecosystems is their structure. To build a quality model, it is necessary to use modeling methods that accurately reflect the properties of software ecosystems. Approaches that characterize them as discrete asynchronous models play an important role in modeling software ecosystems, because these models have all the basic properties: discreteness (a finite number of components, each of which has a finite number of states); the presence of processes and their components, clearly expressed in the phases during which the state changes; publication of information on the completion of each phase of the process; consistency (the beginning of each subsequent phase of the process requires the completion of all previous phases necessary to begin its implementation); parallelism (possibility of simultaneous transitions in several subprocesses and simultaneous execution); asynchrony (no restrictions on the duration of each subprocess and the transition from one phase to another). It is necessary to develop opportunities for modeling real software ecosystems using these approaches, which will create conditions for improving the quality of their design and operation. This article presents an approach to study software ecosystem properties using Petri nets.*

Keywords*: software ecosystem, Petri nets, properties.*

Introduction

The concept of "software ecosystem" is widely used by companies and researchers of the software. According to [Grinenko, 2012] researches of software ecosystems are represented by several definitions depends on their goal, aims, actors etc. That's why throughout this thesis we will consider a few definitions of software ecosystems. So, software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts. The transaction between two software vendors is centred around software

components and software services. In this thesis a definition of software components is used: software component a bundle of software functions accessible through a single interface or carrying a single name which is or can be used as an element in other software packages but of which the core functionality is developed separate from these packages. A software service is a software component accessible via communications outside the users native environment. User can be both human and non-human. Native environment changes depending on the user. If the user is human, the native environment is most probably his or her own computer. If the user is a software package, the native environment consists of the compiled or interpreted code.

Corporation "Microsoft" defines software ecosystem as a set of interactions and mutual influences of organizations (public, educational and commercial) and individuals, which work with software.

Also some authors describe software ecosystems and the typical elements of ecosystems and their context, namely:

- software and its role in IT;
- users of software;
- software creating processes;
- management of the establishment and maintenance of software;
- supply and support of software;
- communication network of state agencies;
- software economy.

According to [Sidorov and Grinenko, 2013] we study the software ecosystems by creating their models. To describe software ecosystems can be used the following tools: i*, UML, Petri nets. Petri nets is the most common nowadays formalism that describes the structure and interaction of parallel systems and processes [Popereshnyak and others, 2019].

According to [Grinenko, 2016] software ecosystems metrics can be the following:

- Productivity. This criterion shows the statistical analysis of nationwide product development and software services for a certain period.
- Diversity. This criterion shows the ranking of IT companies according to the following criteria: "software revenue", "income system integration" and "income from software services."
- Flexibility. This criteria determines which of the fastest growing sectors in IT market.
- Productivity. This criterion determines the impact of investments in the IT industry.

– Sustainability (Vitality). This criterion defines total income in all sectors of the IT market.

– Maturity. This criterion determines the qualitative characteristics of software ecosystems development.

According Petri nets theory these metrics can be analysed using the following structural properties of Petri nets:

– structural liveness
– reachability
– boundedness
– conservatism and partial conservatism
– repeatability and partial repetition.

Usage of Petri Nets to Analyze Software Ecosystem Properties

Petri nets, as well as other methods of modelling systems, are tools of describing the properties of processes of real software ecosystems as asynchronous discrete processes [Kryvyi, 2015]. Like industry standards (UML, BPMN and EMF), Petri nets also describe the basic procedures of discrete graphics processes, including branching (conditional transition, selection), iteration (repetition), and simultaneous (parallel) execution of individual operations or processes. However, the Petri net, in contrast to these standards, mathematically accurately describes the semantics of discrete asynchronous systems, creating the right conditions for their mathematical modeling, analysis and optimization. Simple Petri nets do not have all the properties needed to properly represent the operation of real complex systems. The implementation of simple Petri nets is uncertain: several transitions can be allowed at the same time, and each of them can be implemented. Such a network does not determine the order in which it is executed. Allowed transitions are optional. The allowed transition can be performed or not performed. In other words, a simple Petri net cannot fully automatically reflect the operation of a real system. Simple Petri nets do not consider the concept of transition time, but only indicate the possibility or impossibility of its implementation, so they developed a higher order Petri net, which is also called "time" and "color". In such networks, additional functions (weights) are introduced for positions, transitions and arcs, in particular time [50]. These can be "color" labels, which are perceived only for certain transitions, the time of the transition or the transfer of information by arc, and so on. These networks have greater capabilities for modeling various discrete asynchronous systems, so they have also become widespread.

Mathematical Research of Software Ecosystem Properties

Let us consider SD model (Strategic Dependency model) (fig.1) of software ecosystem. An SD model describes a network of dependency relationships among various actors in an organizational context. The actor is usually identified within the context of the model. This model shows who an actor is and who depend on the work of an actor [2].

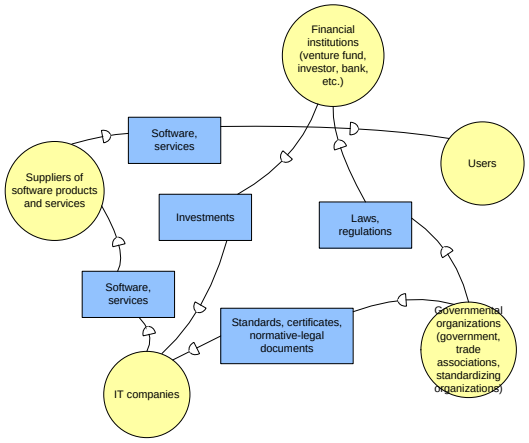

Figure 1. SD model of IT ecosystem

An SD model consists of a set of nodes and links connecting the actors. Nodes represent actors and each link represents a dependency between two actors.

According to SD model it is appropriate for analysis of properties of such system to use transition system, which is defined as

$$TS = (S, T, \alpha, \beta, s_0) \quad (1)$$

where

S is a finite or infinite set of states (places),

T is a finite or infinite set of transitions,

α and β are two mappings from T to S, which gives every $t \in T$ two transitions $\alpha(t)$ and $\beta(t)$, and which are called beginning and end of t transition,

$s_0 \in S$ is an initial state of TS.

As a rule this system is denoted as an oriented graph, vertices of which are states and edges are transitions. In our case and according SD notation, actors' functions and their decomposition are states and their dependency relationships are transitions. Transition system is a kind of Petri nets, that's why all its properties are easy to be researched.

Incidence matrix.

Consider the following order of vertices in the incidence matrix: $u, u_1, u_2, v_2, d_4, w, s_1, d_5, v_1$ (9 places).

Consider the following order of vertices in the incidence matrix:

The order transitions: $u \to u_1, u_2$; $u_1, u_2 \to u$; $u_1 \to v_2$; $v_2 \to u_1$; $v_2 \to d_4$; $d_4 \to v_2$; $d_4 \to w$; $w \to d_4$; $w \to s_1$; $s_1 \to w$; $s_1 \to d_5$; $d_5 \to s_1$; $s_1 \to v_1$; $v_1 \to s_1$ (14 transitions).

To determine the Petri net incidence matrix, we have:

$$a_{ij} = I(t_j, p_i) - I(p_i, t_j),$$

where I is incidence relation of the network. Moreover, given the ordinariness of PN, $\mathrm{I}(\mathrm{x},\mathrm{y}) = 1$, if $(\mathrm{x},\mathrm{y}) \in \mathrm{F}$, and $\mathrm{I}(\mathrm{x},\mathrm{y}) = 0$ otherwise**.**

The incidence matrix (size 9 on 14) looks like:

$$A = \begin{matrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{matrix}$$

1. Boundedness. A PN is structurally limited if it is limited by an arbitrary finite markup. A PN is boundedness if each of its places is limited, that is, for each of its places there is a number *k* such that for arbitrary attainable marking the number of chips in a given vertex will not exceed *k*. Indeed, the top estimate of the number of chips in any vertex is the total number of markup chips that remains constant because the network contains no sources.

We also apply the criterion of structural constraint. According to Theorem 55, MP is structurally bounded if and only if system of linear homogeneous diophantine equations (SLHDE) $A^T y \le 0, x > 0$ has at least one solution over has at least one solution over $\mathbb{N}$. This system is:

$$\begin{gathered} -y_1 + y_2 + y_3 \le 0 \\ y_1 - y_2 - y_3 \le 0 \\ -y_2 + y_4 \le 0 \\ y_2 - y_4 \le 0 \\ -y_4 + y_5 \le 0 \\ y_4 - y_5 \le 0 \\ -y_5 + y_6 \le 0 \\ y_5 - y_6 \le 0 \\ -y_6 + y_7 \le 0 \\ y_6 - y_7 \le 0 \\ -y_7 + y_8 \le 0 \\ y_7 - y_8 \le 0 \\ -y_7 + y_9 \le 0 \\ -y_7 + y_9 \le 0. \end{gathered}$$

If $y_1 = 2$, $y_2 = y_3 = y_4 = y_5 = y_6 = y_7 = y_8 = y_9 = 1$ the left side of all the inequalities rotates to zero, so this is the solution to this SLODN. Therefore, in accordance with the statement of the theorem, this MP is limited.

2. Absence of inaccessible places and dead transitions (Reachability). A place is attainable if, with some attainable layout, the number of chips is positive. With this single-chip Petri net layout, all places are accessible: there is a path for each place that the chip can reach from its original position.

A transition is called dead beyond the markup M unless it is potentially alive beyond that markup. That is, there is no reachable markup from which the transition could work. In this PN, all transitions are potentially alive. That is, for each transition, there is a reach from the initial markup that can trigger it.

3. Liveness. This MP is structurally alive because there is a live network layout. That is, in which at least one transition is possible.

4. Repeatability. This Petri net is repetitive, since for each transition there is a sequence of transitions σ, in which it triggers an infinite number of times.

MP is repeated if and only if SLHDE $\boldsymbol{Ax} \geq \boldsymbol{0}$, $\boldsymbol{x} > 0$ has at least one solution over has at least one solution over $\mathbb{N}$. This system is:

$$\begin{gathered} -x_1 + x_2 \geq 0 \\ x_1 - x_2 - x_3 + x_4 \geq 0 \\ x_1 - x_2 \geq 0 \\ x_3 - x_4 - x_5 + x_6 \geq 0 \\ x_5 - x_6 - x_7 + x_8 \geq 0 \\ x_7 - x_8 - x_9 + x_{10} \geq 0 \\ x_9 - x_{10} - x_{11} + x_{12} - x_{13} + x_{14} \geq 0 \\ x_{11} - x_{12} \geq 0 \\ x_{13} - x_{14} \geq 0. \end{gathered}$$

The solution of this inequality system is

$$x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = x_7 = x_8 = x_9 = x_{10} = x_{11} = x_{12} = x_{13} = x_{14} = 1$$

$$x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = x_7 = x_8 = x_9 = x_{10} = x_{11} = x_{12} = x_{13} = x_{14} = 1,$$

for which the left side of all inequalities is zero.

5. Absence of deadlocks and livelocks. A non-empty set Q, which is a subset of a set of places, is called a deadlock, if $Q^\circ \subseteq {}^\circ Q$.

This PN is all a deadlock, the fragments in it perform the specified relation, although it has no other deadlocks. However, for this PN not to be a deadlock, it is necessary that it has at least one source that contradicts the condition of structural limitations of the PN.

Conclusion

The study demonstrated an ecosystem model and described an approach to the specification of the properties of this model. In this case, the system is labelled TS, which can be studied as Petri net with properties which are defined using this mathematical tool.

Bibliography

[Grinenko, 2012] Grinenko Olena. Software Ecosystems. Scientific Journal of the National Transport University, Vol.36, 2012, pp. 508-512. ISSN 2523-496X (Online), ISSN 2308-6645 (Print) http://publications.ntu.edu.ua/visnyk/26_2_2013/508-512.pdf

[Sidorov and Grinenko, 2013] Sidorov N.A., Grinenko O.O. Software Ecosystem Modelling. Scientific Journal "Software Engineering", Vol.2, 2013, pp. 38-48. ISSN 2306-6512 (Print) http://jrnl.nau.edu.ua/index.php/IPZ

[Popereshnyak and others, 2019] S. Popereshnyak, S. Grinenko, O. Grinenko, O. Kovalenko, T. Radivilova. Methods for Assessing the Maturity Levels of Software Ecosystems. Proceedings of the International Workshop on Cyber Hygiene (CybHyg-2019). Ceur Vol-2654, 2019, pp. 251 - 261. ISSN 1613-0073 (Online) http://ceur-ws.org/Vol-2654/paper20.pdf

[Grinenko, 2016] Grinenko O.O. Metrics of software ecosystems. Proceedings of the International Workshop on Software Engineering, 2016, p. 24.

[Kryvyi, 2015] S.L.Kryvyi. Linear Diophantine constraints and their application. Chernivtsi, Bukrek, 2015.

Authors' Information

Olena Grinenko *– Senior Lecturer of the Software Engineering Department of the National Aviation University. Box: 03058, Kyiv, Ukraine; e-mail: gsa_ck@ukr.net*
Major Fields of Scientific Research: Software Engineering

Sergiy Grinenko *– Assistant of the Software Engineering Department of the National Aviation University. Box: 03058, Kyiv, Ukraine; e-mail: serggrinenko@gmail.com*
Major Fields of Scientific Research: Software Engineering