Designing Software Ecosystems: How Can Modeling Techniques Help?

Mahsa H. Sadi[1], Eric Yu[2,1]

[1]Department of Computer Science, University of Toronto, Canada
[2]Faculty of Information, University of Toronto, Canada
{mhsadi,eric}@cs.toronto.edu

Abstract. It has become an increasingly common practice for software companies to collaborate with external developers in order to develop software platforms for a shared market, constituting software ecosystems. Creating and sustaining a software ecosystem is a challenging problem that involves numerous technical, organizational, and business concerns. To support the systematic design of software ecosystems, modeling is a crucial tool. In this paper, we (a) identify a set of descriptive and analytical requirements raised in the design of software ecosystems; (b) review several modeling techniques used for describing and examining software ecosystems; and (c) assess the support of the reviewed techniques towards addressing the identified requirements. The results provide insight into the gaps between the issues raised in the design of software ecosystems, and the coverage of the studied techniques, suggesting an agenda for future research.

Keywords: Software Ecosystems. Design. Modeling. Analysis. Review.

1 Introduction

Collaboration has become an increasingly critical factor to the success of software companies [1-2]. There are various forces driving software companies to collaborate [3-4]: the shift towards the development of software platforms; the urge to share the costs of production; and the need to satisfy the varying demands of market, which usually fall outside the domain expertise of one software company. These forces have led to a recent software development practice, referred to as software ecosystem in which a keystone software company collaborates with other companies and developers to develop and extend a software platform for a shared market [5-6].

Two well-known examples of software ecosystems are the Google Android and the Apple iOS ecosystems. The main goal of both ecosystems is to provide a software platform, and complementary software applications and services for the market. Hence, both Google and Apple companies have established a network of collaborators (or partners) consisting of application developers, software companies, and content providers. While Google and Apple develop the key software platform (i.e. the mobile operating system), other application developers and software development companies provide complementary software applications and services for the related operating

EMMSAD, p. 1, 2015.

systems. Moreover, content providers supply content such as data, music and game for the applications. Google and Apple make the applications and content developed by external parties visible to the market via the online stores of App Store and Google Play [7-8].

Designing and organizing a sustainable collaboration in software ecosystems is a challenging problem for a keystone software company. First, it involves various boundary decisions, made at the legal and economical borders of the keystone software company and its environment [4, 9]. To open up the software platform and development activities to external companies and developers, the scope of the decisions transcends the organizational boundaries of the company and raise serious concerns and risks about control, ownership, intellectual property, security, privacy, trust, and quality. [9-13]. Second, collaboration in software ecosystems is multi-faceted, spanning various technical, business, and organizational concerns that must be addressed simultaneously [10, 14-15]. A successful software ecosystem needs to have a viable business model, a well-organized inter-organizational interaction model [7, 10], a well-designed collaborative software development process [13], and a software platform that enables the collaboration [16].

For instance, Google and Apple each pursues different approaches to organize their collaboration with external developers and content providers [17]. Each strategy has its own advantages and disadvantages. Google licenses Android for free. This makes the Android platform openly accessible to external developers in order to extend it with complementary applications and services. In contrast, the Apple iOS is proprietary and is accessible to a limited community of software developers directly controlled or owned by Apple. One advantage of Google's strategy in organizing its software ecosystem is that it attracts more software developers and companies to adopt Google Android as a platform. Its disadvantage is that the open strategy results in higher uncertainty in the quality of the final set of software products and makes Android platform and its complementary services and application loosely integrated. On the other hand, Apple's strategy, while limiting the adoption of the platform to a smaller number of developers, leads to a tight integration between platform and its complementary services and applications, and thus a software platform of higher quality [17].

The above example illustrates the complexity of the issues raised in organizing collaboration in a software ecosystem. The pivotal role of a well-organized collaboration among partners in the success software ecosystems [1-3] demands concentrated effort on developing systematic methods and techniques to support the design of software ecosystems. To address this need, a small but growing strand of recent research efforts have specifically focused on providing model-based approaches to describe and analyze software ecosystems. To this objective, two main strategies are pursued: (a) developing new modeling techniques [18], (b) using or adapting the available modeling techniques to describe and analyze software ecosystems [7, 10, 19-21]. However, these modeling approaches vary widely in the terminology that they use, and the analytical capabilities they provide. Moreover, due to the short time since the widespread adoption of the practice of software ecosystem by the software com-

munity, there is as yet no rigorous study on analyzing the needs raised in the design of software ecosystems.

In a preliminary attempt to address the above issues, in this paper, we identify what descriptive and analytical requirements are raised in the design of software ecosystems, and to what extent they are currently supported by a set of modeling techniques used to describe and analyze software ecosystems. The results identify the gaps between the descriptive and analytical needs raised in the design of software ecosystems and the current coverage of model-based approaches, suggesting an agenda for future research.

The rest of this paper is organized as follows: Section 2 identifies a set of descriptive and analytical requirements in designing software ecosystems. Section 3 reviews and summarizes a set of modeling techniques used to examine software ecosystems. Section 4 evaluates the support of the reviewed techniques for the specified requirements. Section 5 concludes the paper and discusses how to improve the support of modeling techniques for designing software ecosystems.

2 Designing Software Ecosystems: A Set of Descriptive and Analytical Requirements

In this section, we identify what issues are raised in the design of software ecosystems and what descriptive and analytical capabilities are required to address these issues. The requirements are developed based on the analysis and synthesis of the available literature on software ecosystems from a design-oriented perspective.

Table 1. A set of requirements to describe software ecosystems

P-1	**Collaborator**: Identifying the members of a software ecosytem and their roles – *Example:* Specifying the keystone software company; content providers; software developers; software companies.	RR*: [10, 18, 22, 23]
P-2	**Interaction**: Identifying the relationships among members – *Example:* Specifying the business or technical relationships among the members of a software ecosystem.	RR*:[10, 18, 22, 23]
P-3	**Activity (or Responsbility):** Specifying the resources, activities and commitment of the members – *Example:*The specific business or software development activity performed by a member.	RR*:[10, 22]
A-1	**Type**:Specifying different types and categories of collaborators, interactions, and responsibilities– *Example:* Identifying a financial or knowledge exchange relationship	RR*: [18, 22]
A-2	**Constraint:** Specifying constraints and rules on collaborators, interactions, and activitiess – *Example:*Describing the conditions and rules on an interaction between a keystone software company and external software developer; or Describing the level of access of one software developer to the platform.	RR*:[18]
A-3	**Attribute:** Specifying attributes and characteristics of collaborators, interactions, and their activities – *Example*: Identifying an important or a reliable collaborator or a critical interaction.	RR*: [18, 24]
A-4	**Characteristic of Collaboration:** Specifying the characteristics and attributes of a collaboration (i.e. the configuration of collaborators, their activities and their interactions) – *Example:* Identifying a healthy, productive or secure collaboration between two or more collaborators.	RR*: [1, 25]

* RR: Related Resources

2.1 A Set of Requirements for Describing Software Ecosystems

The first step in designing software ecosystems is to describe them. The description should provide a clear view of the structure of collaboration or partnership in software ecosystems; i.e., members and the interactions among them. For this purpose, a mod-

eling technique needs to at least be able to describe and represent the following primary elements (either textually or graphically): (a) *Collaborator*; (b) *Interaction*; (c) *Activity (or Responsibility)*. To further delineate the structure of a collaboration, each of the above concepts can be augmented with the following ancillary information: (i) Type; (ii) Constraint; (iii) Attribute; and (iv) Characteristic of Collaboration. The ancillary information enables more elaborate description of a software ecosystem. The description for each of these features is provided in Table 1.

2.2 A Set of Analytical Requirements in Software Ecosystems

To identify what analysis issues are raised in the design of software ecosystems, we adopt a top-down domain analysis approach: we first explain general steps in the development of a software ecosystem; we then identify a set of the analysis concerns raised in each of these steps. A modeling technique used to represent a software ecosystem needs to support answering these analysis concerns.

Table 2. A set of analytical requirements in designing software ecosystems

S-1	**Analyzing incentives and motivations of collaborators**: – *Example questions:* **Q1.** How to foster collaboration and how to motivate external developers and companies to participate and contribute to the platform? / **Q2.** What are the intrinsic and extrinsic motivations of software developers and software companies for joining the software ecosystem? RR*: [17]
S-2	**Analyzing for trust and reliability**: – *Example questions:* **Q1.** How to create and ensure trust between the collaborators? / **Q2.**How reliable is the collaboration? / **Q3.** How reliable are the collaborators? RR*:[6]
O-1	**Analyzing for risk, vulnerability, tolerance, costs and benefits**: – *Example questions:* **Q1.** What risks are involved in the collaboration? / **Q2.** What are the costs and benefits of opening up software platform towards external stakeholders? / **Q3.** What dependencies are created between the collaborators and how critical are these interactions and dependencies? / **Q4.** What if the collaborators do not fulfill their commitments? / **Q5.** How tolerant is the keystone company and other collaborators against potential failures in collaboration? RR*:[12]
O-2	**Analyzing for distributing and decentralizing responsibilities and resources**: – *Example questions*: **Q1.** How to distribute the activities, responsibilities, and resources of software development and service provision among collaborators? RR*: [2, 3, 9, 11]
O-3	**Analyzing for distributing control, authority, decision making, and access**: – *Example questions:* **Q1.** How to distribute the control and authority of decision making over software development among collaborators? / How much control and access should be given to each collaborator over software platform? RR*: [9, 11, 13]
O-4	**Analyzing for distributing ownership and power**: – *Example questions:* **Q1.** How to distribute the ownership of software products and services among collaborators? RR*: [9]
O-5	**Analyzing for openness and sharing in collaboration:** – *Example questions:* **Q1.** What is the acceptable level of openness of the keystone software company in collaboration? / **Q2.** What information, products, and resources need to be shared between the collaborators? / **Q3.** How to open up software development processes and platforms to external collaborators? RR*: [2, 3, 7, 9, 13,15]
O-6	**Analyzing for security and privacy**: – *Example questions:* **Q1.** How to preserve the security and privacy of the platform and processes of the keystone software development organization in collaboration? / **Q2.** Is the collaboration secure? / **Q3.** Is the privacy of collaborators preserved? RR*: [6]
O-7	**Analyzing for health, productivity, robustness, performance**: – *Example questions:* **Q1.** Is the configuration of collaboration productive and robust? / **Q2.** Will the relationships among members lead to a productive collaboration? RR*: [1,2, 9, 25]
O-8	**Analyzing for alignment and conflict resolution**: – *Example questions:* **Q1.** How to resolve conflicts between the collaborators and their contributions to the platform? / **Q2.** How to align the objectives of the collaborators with the keystone software company? RR*: [5, 10]

* RR: Related Resources

Generally, from the perspective of a keystone software company who is in charge of the software platform, three main phases can be considered in the development of a software ecosystem [3, 6]:

1. *Setting up the software ecosystem* **(S)**: The main activities in this step include: (a) to identify the objectives of developing the software ecosystem, and (b) to motivate external stakeholders (including software developers and software companies) to collaborate and contribute to the software platform.
2. *Organizing collaboration and opening up the software development processes and the software platform to collaborators* **(O)**: In this stage, the keystone software company needs (a) to organize and configure the collaboration in the software ecosystem by specifying the collaborators, their roles and activities, and configuring the interactions among them; (b) to decide about how to distribute, decentralize and share access, information, activities, resources, products, responsibilities, and control among the collaborators.
3. *Monitoring and governing the software ecosystem* **(M)**: The main activities in this stage include: (a) monitoring the health and sustainability of the collaboration, (b) orchestrating collaborations among the members, and (c) maintaining and evolving the collaboration and the platform.

In the above phases, specifically phase 1 and 2, several concerns are raised which require elaborate analysis. In Table 2, we identify a set of these concerns by providing example questions that can be raised for a keystone software company. It should be mentioned that the analysis concerns are generic. Therefore, these concerns can be raised for software business managers in the business and organizational context, or for the software project managers and software developers in the software development context.

3 Several Techniques Used for Modeling Software Ecosystems

In this section, we review and summarize a set of modeling techniques that have been used to examine software ecosystems. To include the modeling techniques in the review, two steps have been performed:

Collecting the Modeling Techniques. To gather the modeling techniques, two steps were taken: (a) An extensive search was conducted to collect the available literature on software ecosystems. To perform the search, two recent systemic reviews (published in 2013) [6, 26] were used as an initial catalog for collecting the resources. Then, the collection was updated and extended with more recent literature. The study [6] contains a categorized list of the resources that propose procedures or techniques, qualitative or descriptive models, tools or notations, analytical models, and empirical models for software ecosystems. The study [26] identifies a set of resources on software ecosystem modeling. (b) From the collected set, those research efforts are selected that use a model-based approach and a modeling technique to describe or analyze a software ecosystem.

Selecting the Modeling Techniques. The criteria for the inclusion of the modeling techniques in this review are as follows: **(i)** The collected resource must use a modeling technique to represent the structure of software ecosystems. **(ii)** The members of a

software ecosystem and the relationships among them must be explicitly modeled. **(iii)** The modeling technique must have a well-defined and well-documented syntax, semantics, and notation. The notation can be either graphical or textual.

In the collected literature, a few modeling techniques have been used to describe software ecosystems including Product Deployment Context (PDC) Diagram, a component of Software Ecosystem Meta-model (SEM) [18], Technical Ecosystem Modeling Notation (TECMO) meta-model [27], and UML Deployment Diagram [10], but were omitted according to the second criterion. These techniques focus on modeling the software platform, but do not deal with the involved actors and the relationships among them. Another group of work offers various meta-models such as Associate Models [22], the SPO software ecosystem meta-model [28], and SPEM meta-model [29]. This group is excluded according to the third criterion. The focus of these efforts is on the meta-model level and not on the technique. There were also a few models, such as Graph Representations [30], and Food-web models [8] which lack a well-defined semantics and syntax, and are also omitted from this study.

Ultimately, five modeling techniques were selected that met the above criteria, namely, SSN [18], i* [15, 21], BMC [10, 20], VN [19], and e^{3}Value [7]. From among the selected techniques, only SSN is specifically proposed for modeling software ecosystems. The other techniques are generic and widely used in various domains. In the following, we briefly review how each technique can help describe and examine software ecosystems.

3.1 An Overview of the Selected Techniques

Software Supply Network Diagram (SSN). SSN is one component of the Software Ecosystem Meta-model (SEM), the formally proposed meta-model for describing and analyzing software ecosystems [18]. SSN is used to represent the structures of software supply chains in software ecosystems [24]. SSN explicates the business relationships among the members of a software ecosystem in terms of input and output flows between actors. One specific characteristic of SSN is that its terminology is developed based on the terms used in software development activities. Therefore, it is understandable by software developers.

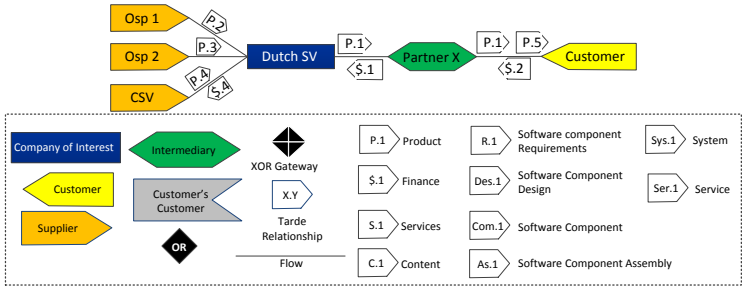

Fig. 1. An example of a SSN Diagram (originally developed in [18])

The main elements of SSN modeling language are "Actors", "Trade Relationships", "Flows", and "Gateways". An "Actor" is an organization or company that participates in a software ecosystem and can be a "Company of Interest", "Supplier",

"Customer", "Intermediatory" or "Customers' customer". A "Trade Relationship" connects two actors, and is comprised of one or more flows. A "Flow" represents an artifact or service from one actor to another and is of different types of: "Products", "Services", "Finance", and "Content". A "Gateway" represents a logical relationship between flows and can be "OR" or "XOR". A SSN diagram is comprised of nodes and edges (see Fig. 1 as an example). Nodes represent the members and their roles in a software ecosystem. Edges represent input / output flows between the members. In SSN, participant actors (organizations) are represented by their names. Trade relationships are depicted in the format of X.Y. X represents the type of flow and Y represents the ID of flow.

The i* Modeling Technique. i* is a generic social modeling technique describing intentional relationships among actors from different business, technical and organizational perspectives [31]. i* explicates the relationships among the members of a software ecosystem in terms of strategic dependencies among strategic actors. The model can be used to explicate the objectives and reasoning of the members for developing or joining a software ecosystem [15, 21].

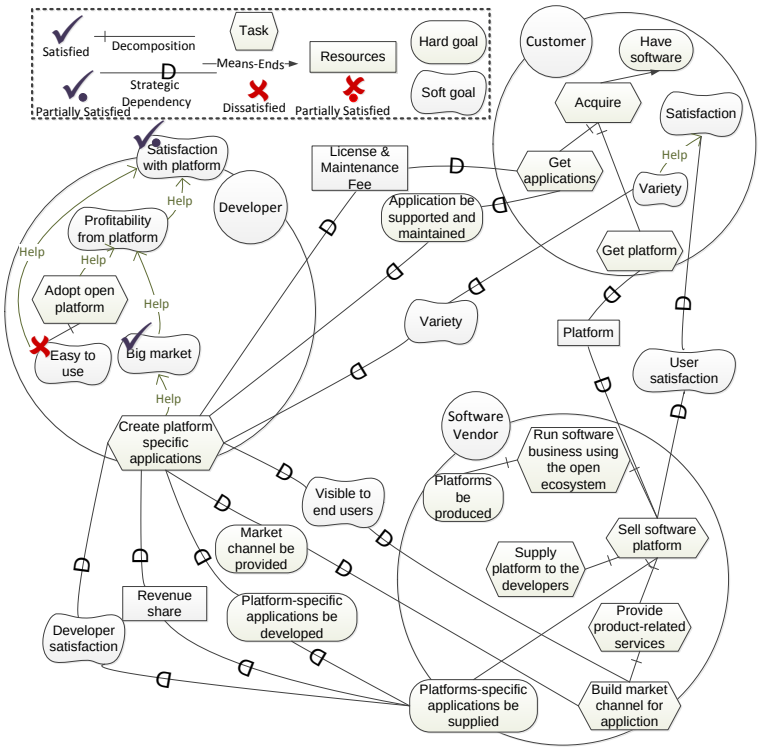

Fig. 2. An example of software ecosystem modeling using i* (excerpted from [21])

In i*, "actors" are intentional, set "Goals" or "Soft Goals", and can come up with different alternatives ("Task", "Resource", "Strategic Dependency") to achieve their goals. Strategic dependencies indicate that one actor relies on another actor to have a goal achieved, a task performed, or a resource furnished. Different types of open,

committed and critical strategic dependencies explicate different degrees of control and vulnerability of the depender in the relationship between two actors. As Fig. 2 illustrates, the motivations of developers for adopting a software platform are captured in terms of soft goals such as "Easy to use", and "Profitability from platform". Moreover, as shown by a sample evaluation of the objectives, although the platform is not easy to use, it adequately satisfies the developers to join the software ecosystem because it has a big market.

Business Model Canvas (BMC). BMC is a structured textual technique, developed based on Business Model Ontology (BMO) – a generic ontology to represent the business model of an organization [32]. This technique can be used to illustrate a high-level business view of a software company and its collaborators, and to describe how a member creates value in a software ecosystem [10, 20] (see Fig. 3 for an example). BMC describes which products and services a software company provides and lists who are its collaborators (partners) in the software ecosystem. For this purpose, the following building blocks are used: The products and services a company provides are listed in the "Value Proposition". The activities and the resources that are necessary to provide the services and products are listed in the "Key Activities" and "Key Resources" section. Customers to whom the organization offers services and products are listed in the "Customer Segments" section. Collaborators and partners of the software company are listed in the "Key Partners" section. "Channels" identifies the means by which the company gets in touch with its customers, and "Relationship" describes the type of link a company establishes between itself and its customer.

<table>
<tr>
<td rowspan="2">Key Partners
• Manufacturer
• Supplier
• Content Provider
• Communication Provider
• Service Operator
• Add-on developer
• Owner
• End user
• Regulatory Agency
• Information Broker</td>
<td>Key Activities
• Production
• Product line management</td>
<td rowspan="2">Value Proposition
Service types:
• Product
• Process
• Life Cycle
• Extended
• Open interfaces
• Quality attributes</td>
<td>Customer Relationships
• Product life cycle
• Liability
• Ownership of information and privacy</td>
<td rowspan="2">Customer Segments
• Broader customer reach
• Improved customer feedback
• Customer differentiation</td>
</tr>
<tr>
<td>Key Resources
• Development Environment
• Human capital
• Branding
• Standards</td>
<td>Channels
• Sales
• Distribution
• Configuration
• Information
• Channel ownership</td>
</tr>
<tr>
<td colspan="3">Cost Structure
• Development cost
• Product cost
• Operating cost
• Information cost</td>
<td colspan="2">Revenue streams
• Volume increases
• Recurring sales
• Direct sales of software
• Subscription fees
• Revenue sharing</td>
</tr>
</table>

Fig. 3. An example of a BMC developed for a software ecosystem (excerpted from [20])

Value Network Diagram (VN). VN is a generic technique to describe the value exchange relationships between a set of human actors. The language of VN is comprised of "Actor", "Transaction", "Value exchange" and "deliverable" concepts, and is mainly used to explicate the business and inter-organizational relationships between a set of organizations [33]. VN can be used to describe and analyze how members create value in a software ecosystem [19] (see Fig. 4 as an example). In VN, the members of a software ecosystem are represented by ovals and the relationships among the members are represented by uni-directional or bi-directional arrows. Arrows represent the exchange of tangible and intangible deliverables (such as goods, services, knowledge, and revenue, or benefit) between the members.

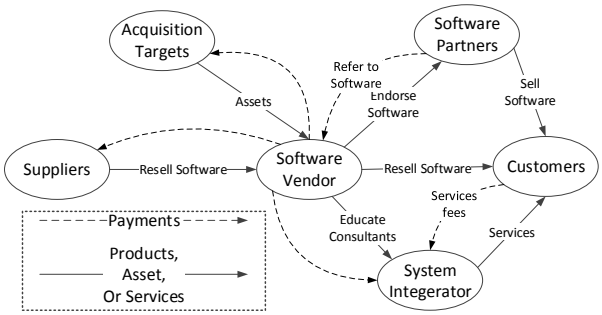

Fig. 4. An example of a VN developed for a software ecosystem (excerpted from [19])

e^{3}Value Modeling. e^{3}Value modeling technique explicates how economic value is created and exchanged within a network of actors [34]. This technique is used to illustrate the economically valuable activities of the members of a software ecosystem and the inter-organizational relationships among them [7] (see Fig. 5 as an example). In e^{3}value modeling, the relationships among the actors are captured in terms of activity flows and input/output flows. The main modeling elements are "Actor", "Market Segment", "Value Activity", "Value exchange", and "Value Object". An Actor" is an economically independent (and often a legal) entity and represents a company or a customer. A group of actors (companies or customers) that share common properties are identified as a "Market Segment". For example, in Fig. 5, "Testing and verification party" is identified as a market segment representing a group of companies which collaborate with the operating system manufacturer to test the operating system. Activities in e^{3}value modeling are "value activities" meaning that they are economically profitable for the actors. Interactions among actor are captured in terms of "Value exchange" and "Value object". "Value Exchange" represents trade relationships between actors. "Value object" represents the exchanged object and can be of the types "Services", "Product", "Money", or "Experiences".

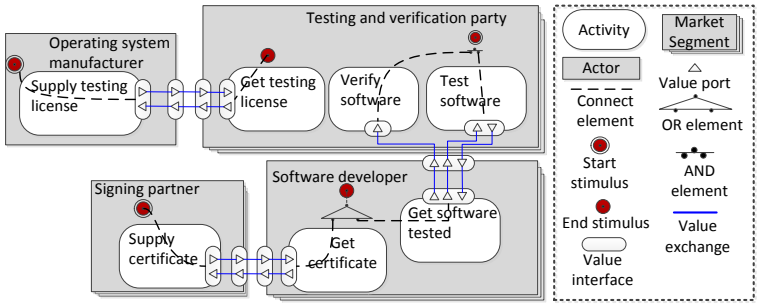

Fig. 5. An example of an e^{3}value developed for a software ecosystem (excerpted from [7])

3.2 Summary of the Reviewed Modeling Techniques

Table 3 provides an overall summary of the reviewed techniques for modeling software ecosystems based on several general criteria. These criteria investigate three main characteristics of the modeling techniques, namely (a) the objective and the

intended users, (b) general usability features, and (c) the specific support and maturity of the techniques for modeling software ecosystems. These characteristics are labeled with A, B and C in Table 3.

Table 3. Summary of the reviewed techniques for modeling software ecosystems

		SSN	i*	VN	BMC	e³V
A-1 Focal viewpoint	Business	+	+	+	+	+
	Inter-organizational	+	+	+	−/− *	+
	Technical (Software Development)	*[1]/+	*/+	−/−	−/−	−/−
A-2 Intended users	Software Business Manager	+	+	+	+	+
	Software Project Manager	−/−	−/+	−/−	−/−	−/*
	Software Developer	−/−	−/+	−/−	−/−	−/−
B-1 Support for analysis	Qualitative	+	+	+	+	+
	Quantitative	−/*	−/×	−/−	−/−	* [2]
B-2 Representation mode	Textual	−	−	−	+	−
	Visual	+	+	+	−	+
B-3 Refinement and traceability		−	+	−	−	−
B-4 Multiple views		* [3]	−	−	−	−
B-5 Formal syntax and semantics		+	+	− [4]	+	+
B-6 Tool support		−	+	−	+	+
C-1 Experimentation maturity for software ecosystems [5]		+	×	×	*	×
C-2 Methodology support for software ecosystems		×	−	−	−	−
C-3 Documentation support for software ecosystems [6]		*	*	×	×	×

Legend: (−): Not supported. (×): Poorly supported. (*): Partially supported. (+): Supported.

* In pair evaluations, the first symbol shows current support of the modeling language and the second symbol shows the potential of the language for supporting the criteria.

Notes:

[1] The main focus of SSN is on modeling the business relationships among the members of a software ecosystem; however, the terminology is based on software development activities and can support the technical relationships to some extent.

[2] e³V supports quantitative analysis on the financial aspect of interactions among actors.

[3] Although SSN does not support multiple views of a software ecosystem, it is accompanied by another component in the SEM meta-model, named Product Deployment Context (PDC). PDC provides the architectural viewpoint of a software ecosystem.

[4] VN lacks a formal definition of syntax. It is not clearly defined what information should be represented in the models and how the information should be represented.

[5] This criterion is assessed as follows: (×) Poorly supported: the model is experimented in examples. (*) Partially supported: the model is experimented in real-world case studies by researchers. (+) Supported: the model is experimented by practitionars and intended users in real settings.

[6] This criterion is assessed as follows: (×) Poorly supported: the number of available documentation; i.e. publications, technical papers and websites is less than 2. (*) Partially supported: the number of available documentation is more than 2 and less than 5. (+) Supported: the number of available documentation is more than 5.

As the criteria of group A demonstrate, the focus of the modeling techniques is mainly on addressing the business and inter-organizational aspects of collaboration in software ecosystems, reflecting the viewpoint of software business managers. Group B criteria show that all of the techniques mainly support qualitative analysis, and the majority of the techniques do not support or poorly support two features of “refinement and traceability”, and “multiple views”. Refinement criterion evaluates the support of the modeling technique for developing a hierarchy of models with different levels of details and different levels of information in them and the ability to trace between the models. The multiple views criterion evaluates the ability of the technique to represent different views from a software ecosystem for different stakehold-

ers. Finally, the criteria of group C reveal that currently, the majority of modeling techniques have not received adequate experimentation in modeling software ecosystems and there is not yet enough documentation, methodology, or guidelines available for using these techniques in the practice of software ecosystems.

4 Applying the Requirements to Assess the Reviewed Techniques

In this section, we analyze and assess the nature and extent of support offered by the reviewed modeling techniques towards the design of software ecosystems based on the descriptive and analytical requirements identified in Section 2.

4.1 Support for Describing Software Ecosystems

Evaluation Procedure. To evaluate the support of each modeling technique, it has been checked whether the technique supports representing the primary and ancillary elements, (P-1 to P-3) and (A-1 to A-4), introduced in Table 1.

Evaluation Results. The assessment of the reviewed modeling techniques based on the descriptive requirements is presented in Table 4. As Table 4 demonstrates, except for BMC, all the modeling techniques support representing collaborators and their interactions in software ecosystems. In all the studied techniques, collaborators are represented in terms of actors (but in each technique, actor has a different meaning.), and the interactions among collaborators are mostly represented in terms of input-output flows. However, the notion of activity is not supported by most of the modeling techniques. Moreover, the majority of techniques do not support describing ancillary information about the collaborators, the activities and the interactions among them.

Table 4. Assessment of the modeling techniques for describing software ecosystems

	P-1: Collaborator			P-2: Activity (or Responsibility)			P-3:Interaction			A-4
	A-1	A-2	A-3	A-1	A-2	A-3	A-1	A-2	A-3	
SSN	+/(Actor)*			– [1]			+/(I-O Flow)			–
	+ **	–	+	–	–	–	+	–	–	
i*	+/(Actor/ Role/Agent)			+/(Tasks/Resources/Goals/ Softgoals)			+ /(Strategic Dependency)			–
	–	–	–	–	–	–	–	–	+	
VN	+/(Actor)			–			+/(Activity Flow)			–
	–	–	–	–	–	–	+	–	–	
BMC	+/(Actor)			* [2] /(Activities / Resources)			–			–
	–	–	–	–	–	–	–	–	–	
e^{3}V	+ / (Market Segment /Actor)			+ (Activities)			+/(I-O Flow/Activity Flow)			–
	–	–	–	–	–	–	+	–	–	

Legend: (−): Not supported. (*): Partially supported. (+): Supported.

* The first row in front of each modeling technique shows the support for describing the primary elements . The related element of the modeling technique that support the represnetation of the primary element is also identified.

** The second row in front of each modeling technique shows the support for describing the anciliary concepts.

Notes:

[1] SSN does not support representing the repsonbilities of collaborators. However, the other component of SEM, Product Deployment Context (PDC), identifies the architectural components of a software ecosystem and SEM enables linking the actors in SSN to the relevant components in PDC. [18]

[2] BMC only identifies the key activities and resources of one colalborator (software company).

4.2 Support for Analyzing Software Ecosystems

Evaluation Procedure. To evaluate the support of the modeling techniques for each type of analysis identified in Table 2, the following four criteria are used:
(1) Information Support (IS). This criterion evaluates whether the model expresses the required information to draw conclusion about an analysis concern.
(2) Analysis Representation (AR). This criterion identifies whether the information related to the analysis is captured inside the model or outside the model.
(3) Alternative Analysis and Comparison (AAC). This criterion identifies whether the modeling technique enables representing and comparing the consequences of two or more alternatives to address one analysis concern.
(4) Type of Analysis (TA). This criterion identifies whether the modeling technique enables descriptive analysis or predictive analysis.
Evaluation Results. Table 5 evaluates the capabilities of the reviewed modeling techniques to support the analytical requirements of software ecosystems (S-1 to O-8 in Table 2). As the results demonstrate, the majority of the reviewed techniques do not provide enough information (IS) to address the analysis questions raised in the design of a software ecosystems. From among those techniques that provide adequate information support for one type analysis, the majority do not capture adequate information related to performing the analysis (AR). Representing and comparing alternatives to address the analysis concerns (AAC) is covered by only one technique (i*). Finally, all of the models merely enable descriptive analysis and do not support prediction and prescription capabilities for designing a software ecosystem.

Table 5. Assessment of the modeling techniques for analyzing software ecosystems

	S-1	**S-2**	**O-1**	**O-2**	**O-3**	**O-4**	**O-5**	**O-6**	**O-7**	**O-8**
SSN	−/O *	−/O	+/O	−/O	−/O	−/O	+/O	−/O	−/O	−/O
	O/D **									
i*	+/I	+/I	+/I	+/I	+/I	−/I	+/I	+/I	−/I	+/I
	I/D									
VN	−/O	−/O	+/O	−/O	−/O	−/O	+/O	−/O	−/O	−/O
	O/D									
BMC	−/O	−/O	−/O	−/O	−/O	−/O	−/O	−/O	−/O	−/O
	O/D									
e^3V	−/O	−/O	+/O	+/O	+/O	+/O	+/O	−/O	−/O	−/O
	O/D									

Legend: (−): Unable to support. (+): Able to support. I: Information captured inside the model. O: Information captured outside the model. D: Descriptive analysis. P: Predictive or Prescriptive analysis.

* The pair evaluations in the first row in front of each modeling technique identify the following information: The first symbol evaluates the *information support*. The second symbol evaluates *analysis representation capability*.

** The pair evaluations in the second row in front of each modeling technique identify the following information: The first symbol evaluates *alternative analysis and comparison* capability. The second symbol identifies the *type of analysis* supported.

5 Conclusion

Software ecosystem is a recent software development practice in which various software companies, application developers, and content providers collaborate to develop software platforms, and complementary software applications and services for a shared market. Herein, we identified a set of descriptive and analytical requirements raised in the design of software ecosystems, and investigated to what extent these requirements are addressed by a set of techniques used to model software ecosystems. In the following, we identify the gaps and suggest how to enhance the modeling support for designing software ecosystems:

- *Lack of support for representing the technical aspect of collaboration*: Designing and organizing a sustainable collaboration in software ecosystems involves technical concerns as well as business concerns. However, the focus of the studied modeling techniques is mainly on describing and analyzing the business aspect of collaboration. They mainly reflect the viewpoint of software business managers. Modeling techniques that reflect the viewpoint of software project managers and software developers, and the technical relationships among the members can complement the reviewed techniques.
- *Lack of alignment between the business and organizational viewpoints and the technical viewpoints*: Technical aspect of collaborations in a software ecosystem should follow the rules and restrictions in the business and organizational aspects [6]. Models are the main tools for aligning and tracing between these dimensions. Specifically, two features of "refinement and traceability", and "multiple views" in modeling techniques enable aligning between different viewpoints. Enriching these features in the studied modeling techniques alleviates the issue of alignment in designing software ecosystems.
- *Weak representation support*: SSN is the formally proposed technique for modeling the relationships among the members of a software ecosystem. However, SSN does not support describing the activities of the collaborators in software ecosystems. Moreover, the majority of the studied techniques provide very little support for describing constraints and the attributes of collaborators, their activities, and the interactions among them. Enriching the syntax and semantics of the studied techniques to support these features, or using the techniques that already support these features enhance the representation of software ecosystems.
- *Weak methodological support*: There is not enough methodology and documentation support on how to model and analyze software ecosystems using the reviewed techniques. Moreover, most of the studied modeling techniques have not received enough experimentation and evaluation in real case studies and by practitioners. Addressing these weaknesses leads to further clarification of the descriptive and analytical needs in software ecosystems as well as how to improve the effectiveness of each technique.
- *Weak analysis support*: The majority of the studied techniques do not include enough information to support the analysis concerns raised in the design of software ecosystems. Alternative analysis and comparison is not supported by most of the modeling techniques. No support is provided for the predictive and prescriptive analysis on software ecosystems. Enriching the support of each individual technique for analysis, or providing guidelines to use a group of the modeling techniques in combi-

nation facilitates the design and development of software ecosystems.

Limitations of the Study. In this study, an initial set of modeling requirements for designing software ecosystems is identified through the analysis of the published literature. Confirming these requirements with practitioners and/or other active research groups in software ecosystems strengthens the results of this study. Moreover, the assessment of the modeling techniques against the identified analytical requirements needs to be further complemented by elaborating the evaluation criteria and conducting empirical studies on the modeling techniques.

References

1. Iansiti, M., & Levien, R. (2004). Strategy as ecology. Harvard business review, 82(3), 68-81.
2. Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. In 31st International Conference on Software Engineering-Companion Volume, 2009 (pp. 187-190).
3. Bosch, J. (2009). From software product lines to software ecosystems. In Proceedings of the 13th International Software Product Line Conference (pp. 111-119). Carnegie Mellon University.
4. Bosch, J. (2012). Software ecosystems: Taking software development beyond the boundaries of the organization. Journal of Systems and Software, 85(7), 1453-1454.
5. Jansen, S., & Cusumano, M. (2012). M.: Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance. In Proceedings of fourth international Workshop on Software Ecosystems (pp. 41-58).
6. Manikas, K., & Hansen, K. M. (2013). Software ecosystems–a systematic literature review. *Journal of Systems and Software*, *86*(5), 1294-1306.
7. Müller, R. M., Kijl, B., & Martens, J. K. (2011). A comparison of inter-organizational business models of mobile app stores: there is more than open vs. closed. *Journal of theoretical and applied electronic commerce research*,*6*(2), 63-76.
8. Lin, F., & Ye, W. (2009)**.** Operating system battle in the ecosystem of smartphone industry. In *International Symposium on Information Engineering and Electronic Commerce, 2009. IEEC'09.* (pp. 617-621).
9. Jansen, S., Brinkkemper, S., Souer, J., & Luinenburg, L. (2012). Shades of gray: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, *85*(7), 1495-1510.
10. Christensen, H. B., Hansen, K. M., Kyng, M., & Manikas, K. (2014). Analysis and design of software ecosystem architectures–Towards the 4S telemedicine ecosystem. Information and Software Technology, 56(11), 1476-1492.
11. Boudreau, K. (2010). Open platform strategies and innovation: Granting access vs. devolving control. Management Science, 56(10), 1849-1872.
12. Franch, X., Susi, A., Annosi, M. C., Ayala, C. P., Glott, R., Gross, D.,& Siena, A. (2013). Managing Risk in Open Source Software Adoption. In ICSOFT (pp. 258-264).
13. Hanssen, G. K. (2012). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of Systems and Software, 85(7), 1455-1466.
14. Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., & Robinson, W. (2011). The brave new world of design requirements. Information Systems, 36(7), 992-1008.
15. Sadi, M. H., & Yu, E. (2014). Analyzing the evolution of software development: From creative chaos to software ecosystems. In IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), (pp. 1-11). IEEE.

16. Cataldo, M., & Herbsleb, J. D. (2010). Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (pp. 65-72). ACM.
17. Koch, S., & Kerschbaum, M. (2014). Joining a smartphone ecosystem: Application developers' motivations and decision criteria. *Information and Software Technology*, 56(11), 1423-1435.
18. Boucharas, V., Jansen, S., and Brinkkemper, S. (2009): "Formalizing software ecosystem modeling." Proceedings of the 1st international workshop on Open component ecosystems.
19. Popp, K., & Meyer, R. (2010). Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry. Books on Demand.
20. Axelsson, J., Papatheocharous, E., & Andersson, J. (2014). Characteristics of software ecosystems for federated embedded systems: A case study. Information and Software Technology , 56(11), 1457-1475.
21. Yu, E., & Deng, S. (2011). Understanding software ecosystems: A strategic modeling approach. proc of 3rd IWSECO, 65-76.
22. van Angeren, J., Kabbedijk, J., Jansen, S., & Popp, K. M. (2011). A Survey of Associate Models used within Large Software Ecosystems. In *IWSECO@ ICSOB*, (pp. 27-39).
23. Jansen, S., Brinkkemper, S., & Finkelstein, A. (2009). Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. In IWSECO@ ICSR.
24. Handoyo, E., Jansen, S., & Brinkkemper, S. (2013). Software ecosystem modeling: the value chains. In Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems (pp. 17-24).
25. van den Berk, I., Jansen, S., & Luinenburg, L. (2010). Software ecosystems: a software ecosystem strategy assessment model. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume* (pp. 127-134).
26. Werner, C., & Jansen, S. (2013). A systematic mapping study on software ecosystems from a three-dimensional perspective. Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, 59-81.
27. Seidl, C., & Aßmann, U. (2013). Towards modeling and analyzing variability in evolving software ecosystems. In Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems (p. 3).
28. Lungu, M., Lanza, M., Gîrba, T., & Robbes, R. (2010). The small project observatory: Visualizing software ecosystems. Science of Computer Programming, 75(4), 264-275.
29. Pettersson, O., Svensson, M., Gil, D., Andersson, J., & Milrad, M. (2010). On the role of software process modeling in software ecosystem design. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (pp. 103-110).
30. Manikas, K., & Hansen, K. M. (2013). Characterizing the Danish telemedicine ecosystem: making sense of actor relationships. In Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems (pp. 211-218).
31. Yu, E., Giorgini, P., Maiden, N., & Mylopolous, J. (Eds.). (2011). Social modeling for requirements engineering. MIT Press.
32. Osterwalder, A. (2004). The business model ontology: A proposition in a design science approach. Ph. D. Dissertation. Institut d'Informatique et Organisation. Lausanne, Switzerland, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC, 173.
33. Allee, V. (2008). Value network analysis and value conversion of tangible and intangible assets. Journal of Intellectual Capital, 9(1), 5- 24.
34. Gordijn, J., Akkermans, H., & Van Vliet, H. (2000). Business modelling is not process modelling. 1st Workshop on Conceptual modeling for e-business and the web (pp. 40-51).