Competition and Collaboration in Requirements Engineering: A Case Study of an Emerging Software Ecosystem

George Valença
Department of Statistics and Informatics
Federal Rural University of Pernambuco
Pernambuco, Brazil
Email: georgevalenca@deinfo.ufrpe.br

Carina Alves[1,2], Virgínia Heimann
Informatics Center[1]
Federal University of Pernambuco
Pernambuco, Brazil
Email: {cfa, vmch}@cin.ufpe.br

Slinger Jansen, Sjaak Brinkkemper
Department of Information
and Computing Sciences[2]
Utrecht University
Utrecht, the Netherlands
Email: {s.jansen, s.brinkkemper}@cs.uu.nl

***Abstract*—Increasingly, small to medium software producing organisations are working together in collaboration networks to supply complex compositions of their products and services to customers. In this paper, we present a case study of two software companies that are evolving their partnership towards the creation of a software ecosystem. We investigate the impacts of their tightening partnership on software product management, with a focus on requirements engineering practices. We observe that the requirements definition and negotiation processes are directly affected by their fluid collaborative and competitive relationships. Power disputes, volatile roles and mismatches in release synchronisation are also aspects observed in the studied software ecosystem. We extract several observations from the case study that support small to medium software firms in making decisions within their software ecosystem.**

I. Introduction

The changing dynamics of the globalised software industry have influenced companies to operate in a complex and networked setting. Software companies are increasingly redefining roles and patterns of cooperation and innovation according to their position in a value chain [1]. When competing in the same market, companies need to interact to survive in a turbulent environment. This emerging concept is called Software Ecosystem (SECO), which is defined as "a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them" [2]. Software ecosystems differ from global software engineering, where development is dispersed in several organisations that do not own the software produced and interaction among them is rather transitory. In a software ecosystem setting, a common technology or platform binds the software products. SECOs are based on the notion of organisational ecology, with a radically different business model where frequent interaction and value sharing between players promote self-regulation in the network [3].

Yu [4] suggests that for an ecosystem to operate healthy, an energy source is necessary. In a biological ecosystem, the sun is the energy source. In a software ecosystem, the market is the main source of energy that can directly and indirectly affect all players of the ecosystem. By establishing collaborative and complementary relationships with suppliers and customers, software companies participating in a SECO can cope with financial, time and knowledge constraints [5]. Moreover, they can co-evolve in a hub of local or global market, complementing features and dividing R&D costs [6].

Software ecosystems introduce new challenges in business, social and technical levels [7]. Companies interacting in a SECO shall trigger efforts towards an increasing platformisation of their products for outside development, via extensible component-based architectures [6]. Software maintenance and evolution are important issues to be addressed in software ecosystems. For instance, players have to manage the risks of reusing components with potential low technical quality [8]. The interaction among partners also raises the need for knowledge management to support communication and efficient propagation of information [2]. Since products features are provided by different partners in an ecosystem, requirements definition and negotiation are also affected by their specific interests [9]. Software ecosystems provide products for a wide market, which means that requirements are often jointly defined by the suppliers of the ecosystem rather than elicited from users. However, actors may have conflicting priorities that have to be reconciled. For instance, individual requirements must be intertwined and merged into a set of complementary features. This increasingly networked structure requires new business models to address issues of ownership and open innovation.

Several studies have investigated software ecosystems from various domains, such as open source [10], app stores [11] and online games [12]. However, relatively little scientific knowledge is available regarding the dynamics of an emerging software ecosystem composed by Small and Medium Enterprises (SMEs). The major strength of SMEs is often their flexibility and adaptability. They are often more skilled than larger companies in adapting to new customer requirements and adopting innovative methods. On the other hand, SMEs usually have limited financial and human resources. To strengthen their market presence, growth strategies include establishing joint ventures and alliances.

In this paper, we investigate how SMEs already engaged in collaborative networks evolve their products towards a software ecosystem. We aim to understand how software product management and requirements engineering, in particular, are conducted by partner companies. From an industrial perspective, this study aims at understanding the context and challenges faced by SMEs willing to create a healthy software ecosystem. We conducted a case study of two Brazilian software suppliers that are creating an ecosystem with other partners.

This paper is structured as follows. In section 2, we present the research method adopted, which includes the research questions, data collection and analysis procedures. Section 3 presents the results from the case study. In Section 4, we discuss a set of observations that synthesise important issues faced by the studied companies in the context of their software ecosystem. We also analyse these propositions in light of related literature in the field. Finally, in Section 5 we present conclusions and future work.

II. Method

The focus of this case study has been to investigate how SMEs evolve their relationships to establish a software ecosystem around their products. We refined this objective in the following research questions (RQ):

- RQ1 - How is the process of software product management adjusted to cope with the challenges introduced by tighter collaboration with an ecosystem partner?
- RQ2 - What are the implications for requirements engineering in a software ecosystem composed by SMEs?
- RQ3 - How do partnerships evolve among SMEs towards the creation of a software ecosystem?

To answer the research questions, we conducted a case study of two small to medium software companies that are forming a software ecosystem. We adopted an information-oriented selection and purposively chose companies that were representative to our study on the basis of expectations about their information content. An interpretative case study supports our primarily exploratory and descriptive purpose of comprehending a particular situation through participants interpretation of their context [13].

A. Data Gathering and Analysis

Empirical data was collected through open-ended and semi-structured interviews. We defined an interview protocol with thirty-five questions that were directly refined from the research questions. This instrument ensured that the same basic structure was followed in each interview. However, new questions could be brought up according to interviewees discourse as a way to provide a more in-depth understanding of the investigated topics. Based on the protocol, we defined the sampling of interview participants. The research questions required the coverage of technical and managerial roles from both companies. Using a snowball sampling strategy, interviewees were asked to recommend other participants based on their expertise in the field.

The interviews were carried out with eleven professionals. Each interview lasted from 20 to 70 minutes. During each one, we agreed upon a nondisclosure agreement (NDA) to handle data anonymously. Participants were asked to explain company's internal documents such as requirements specification and project plans. These artefacts complemented the interviews and were considered additional sources of evidence. We further contacted participants by e-mail to clarify and extend data gathered during the interviews. Table I provides an overview of the participants.

TABLE I
Participants

Company	Job Function	Years at the company
A	Project Manager	5 years
	Business Analyst	3 years
	System Analyst	5 years
B	Project Manager	6 years
	Product Manager	11 years
	Release Manager	4 years
	Integ. Team Leader	1,5 year
	Business Analyst	6 years
	System Analyst 1	2 years
	System Analyst 2	3 years
	Tester	3 years

The different number of participants interviewed per company was due to the fact that professionals from Company A play several roles. For instance, the project manager also acts as product manager and product owner during requirements prioritisation. The system analyst is also responsible for software development and testing. In its turn, Company B separates these functions in specific roles, such as the release manager and the integration team leader.

All interviews were recorded and further transcribed by two researchers. With the support of Weft QDA software tool for qualitative research [14], we analysed interview transcripts through a coding procedure. We classified related quotations in labels and subsequently assembled them in categories. Then, we examined facts, concepts, ideas and their relationships. Based on these results, we derived a set of observations to explain how studied companies are forming a software ecosystem, and how the software product management and requirements engineering practices are affected.

B. Case Companies

The case study reported in this paper results from the investigation of two software organisations situated in Recife/Brazil. Company A produces software products for retail chains, distributors and wholesalers markets. It has five software products in the portfolio. The company was founded in 1986, and currently has 70 employees. In 2013, the company has achieved level F at the Brazilian Software Process Improvement Program (MPS.BR) [15]. This program is inspired on CMMI process improvement approach. In the case study, we explored software integrations around the company's main commercial product. The product is an ERP that automates business rules of distribution, wholesale and retail network,

focusing on the management of suppliers and products for resale. This product is used by more than 2.000 registered users from 12 customers in Brazil and abroad. In addition, the company provides system deployment and bespoke development services to specific customers.

Company B is specialised in developing management systems and providing consulting services. Its main product is a complete ERP solution that provides business automation for several market niches, such as: health, oil and gas, sugar industry and logistics. The product's main strategic differential is the module of accounting and tax compliance that is automatically updated with the frequent changes of Brazilian accounting standards. The company has around 300 employees distributed in its headquarter in Recife and branch offices in Maceio and Sao Paulo. In the last years, the company has established a leadership position within regional IT industry, which results from a portfolio of 15 software products. The company has a strong focus on software quality, having achieved the following certifications: ISO9001, CMMI level 2 and MPS.BR level C. We investigated the context of its main software product, an ERP which currently is adopted by 16.000 users from 600 customers in Brazil and abroad.

III. Results

In this section we present the results obtained from our study in Companies A and B. The subsections correspond to the categories that emerged from the case study analysis.

A. Portfolio Management

The product investigated at Company A provides enterprise procurement solution for retail chains, distributors and wholesalers markets. In particular, it provides stock supply, inventory management and tax review functionalities. Company B serves the market with a full ERP focused on accounting and finance areas. Accounting parameterisation is a distinguishing feature provided by this system, which allows users to fully configure any accounting entry. In both companies, product integrations can occur in the following ways: between products from partners, with bespoke systems from customers or with products from other suppliers that serve a particular customer. Currently, the product from Company A is integrated with 15 other products, while the product from Company B has 110 product integrations. The studied companies have partnerships with local, national and international companies.

To better structure upcoming releases and increase customer satisfaction, Company A decided to focus on specific features of the product. Finance and procurement were prioritised over tax and accounting functionalities. The company modularised the system and started to establish integration partnerships with other suppliers. According to the company's project manager, *"partners are experts on features that the product does not provide or features that were discontinued"*. By complementing each others functionalities, partners support the co-evolution of their products. This trend is reinforced by the product manager of Company B: *"we establish a partnership for specific parts of the system that we have no intention to develop"*. To select products and respective partners, this company defines a set of criteria that partners should satisfy. In particular, they look for partners that allow the company to enter in new vertical markets.

The companies generally initiate a partnership when facing challenges in a new market segment, as exemplified by the project manager of Company B: *"Since we do not have enough experience in the retail segment, we are having problems to cope with the flow of innovations in the field"*. Therefore, Company B proposed the collaboration with Company A as a strategy to enter in the retail market. For Company A, the main motivation to establish the partnership is due to the superior accounting features of Company's B product.

The products are commercialised under different contract types. Customers may acquire the product from Company A by renting it for a fixed price during a pre-defined period, or through a project involving consultancy and a monthly fee that is annually adjusted. Company B provides two contract models: a monthly fee that is calculated according to the number of licenses rented and the customer size, or a reduced monthly fee to cover maintenance costs.

B. Product Roadmapping

Product roadmaps in the case companies typically cover one year. At Company A, the product committee defines the product roadmap by analysing potential market demands and new features to fulfill these needs. The project manager from this company noted that the product roadmap is a *"market demand to define a proactive vision of the product improvements"*. In the beginning of every year, customers require the product roadmap to understand what releases are planned and how the product will evolve. In some cases, roadmaps can be developed in close collaboration with customers to create customer-specific product roadmaps. These roadmaps establish a commitment between both parties and can be used as a basis to negotiate the integration contract. The project manager added that, depending on top management decisions, information about specific features can be retained to increase product value.

The product manager at Company B develops the product roadmap aligned with the company's strategic planning and based on customers' feedbacks. According to him, to enter in a new market segment *"the product must be adapted to speak its language... Customers want to know how the product features fulfill the business processes of their segment, i.e. which features will enter in the product roadmap, in which order, at what time, etc. After I understand the domain and the products that support this segment, I start to propose new features for the product"*.

Currently, Company B faces challenges in balancing resources to develop customer-specific customisations and ensure the evolution of the product according to the planned roadmap. To address this situation, the development teams have been assigned to dedicate more time developing new features from the roadmap than satisfying specific demands from customers. This prioritisation has been aligned with the

company's strategic plans to target new markets. This means that the product evolution aims to innovate the platform rather than adding customer-specific features into the product.

C. Release Planning

Both companies define the release planning aligned with their product roadmap. At Company A, the project manager and members of the development team form a product committee to plan the releases. At Company B, the product manager is responsible for the definition of next features to be launched. She arranges meetings with the business analyst to detail each new feature and provide development guidance for developers. Interviewees stressed that quite frequently emergent demands from customers can change the release planning.

Product releases are of two types in the case companies. Regular releases can address legal issues and provide novel functionalities. These new features are part of the product roadmap or originate from customer-specific customisations that companies consider relevant to include in the standard version of the products. In addition, frequent bug fix releases are detailed in release notes sent to all customers. At Company A, there is a regular product release every six months. Release frequency in Company B has been gradually increased in recent years. Latest versions were launched every month to speed time-to-market. However, this strategy created several problems:

- The short release cycle increased the number of bugs in the product. It was required to release several intermediate versions to fix bugs;
- Customers considered the steady flow of new versions disturbing, since updating the product frequently may represent a risk for their running systems;
- The company adopted a reactive approach by prioritising customer-specific customisations and neglecting the implementation of new features from the product roadmap;
- The very short release schedules impacted the routine of team members, who were under strong pressure to package new releases every month. Since team sizes remained the same, the company was demanding unrealistic efficiency from the teams.

After recognising all the issues caused by the short release cycles, Company B started to launch new product releases every two months. The current goal is to adopt a biannual periodicity for regular releases and launch additional versions for specific customers every two months. During integration projects, partner companies must deploy their products simultaneously at customer site. However, release planning cycles are not properly synchronised and product versions are launched in different moments. Development teams from both companies lack a centralised communication and they are not coordinated to treat integration conflicts. In particular, although a project manager from a given partner may be responsible for the integration project, he has no authority over the development team of the other company. Therefore, new product versions can originate several mismatches in the product integrations. Fixing these problems is frequently postponed due to limited team resources from the companies who are exclusively dedicated to integration projects. The project manager of Company A raised an additional problem: *"my need or urgency may not be the same of my partner"*. The companies are aware of these challenges. They currently try to minimise integration problems through follow-up meetings conducted by project managers, who aim at improving the coordination of partners development teams and synchronising developments' prioritisation. However, the companies do not have a clear strategy on how to align their release planning. This situation limits the co-evolution of their products.

D. Requirements Engineering

As market-driven software suppliers, the case companies must understand the needs of market segments to define the product vision and identify market requirements. In Figure 1 we describe the external relationships of the teams from both companies. It evidences the market-driven and customer-specific requirements engineering processes performed by the case companies. It also highlights that several interactions occur between the teams during different phases of the requirements engineering process. This means that partners working in a software ecosystem should make joint decisions regarding the requirements for their products.

At Company A, the product committee and experts in the business domain define the overall product evolution strategy. Moreover, the committee evaluates if requirements defined for a specific integration project can be suitable for a larger group of customers. At Company B, the product manager defines a set of product features in the roadmap to satisfy strategic demands related to current customer market segments as well as targeted market niches. During integration projets, requirements elicitation aims at identifying customisation requests from customers, integration needs from partners, and legal demands. At Company A, requirements elicitation is carried out either by the project manager or business analyst, while the project manager is responsible for this activity at Company B. Company A created an artefact called 'Analysis of Requests and Needs' (ARN), which provides a questionnaire for customers to describe their demands. However, the discourse of the business analyst evidenced flaws in the elicitation process: *"after the system is deployed, the customer sometimes reports that there is something in his process that he did not state; something that was not identified during the elicitation can emerge as a need. This new demand can be analysed and implemented in future releases along the year"*.

Companies receive demands from partners to develop new integrations among the products. Integration requirements are jointly defined by team members. These requirements involve technical aspects such as database tables and mappings among systems, data dictionary, data types exchanged and integration flows. Integration requirements must be *"generic and reusable"* among partners, stated a system analyst at Company B. Integration and functional requirements are separated in the project documentation. Interviewees argued that using such simple categorisation accelerates requirements documentation.

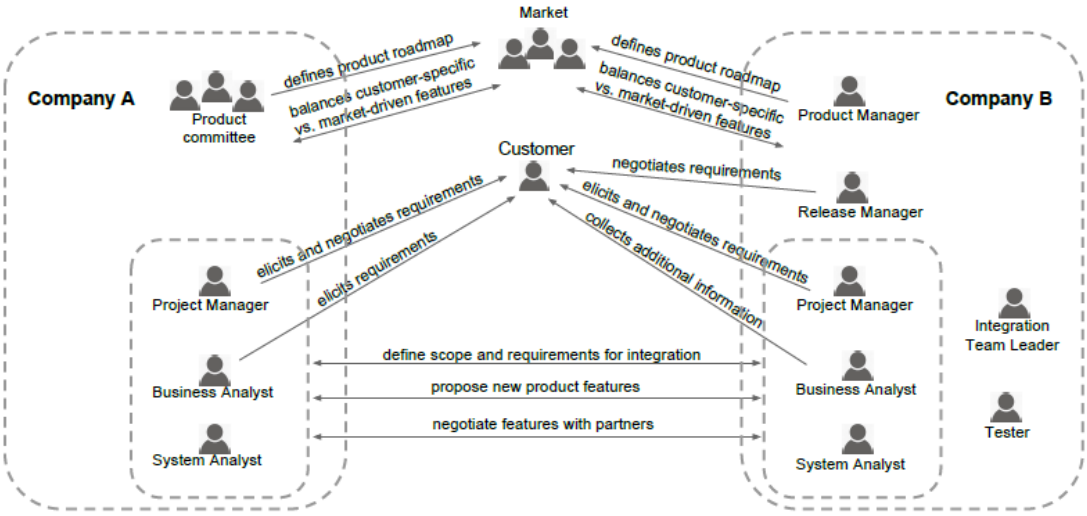

Fig. 1. External Interactions during the Requirements Engineering Process.

A system analyst at Company B explained that integration requirements are identified through functional requirements: *"I need to know the functional requirement to see what is related to the integration. For instance, to create a field in the product registration form I need to define how integrating systems will access it"*. The companies do not formally specify quality requirements, which are simply treated during the development process to address performance and security issues, for instance.

Additional sources of requirements include new ideas provided by project managers, development teams, business analysts, management unities and even the companies' presidents. Once an integration project is concluded, stakeholders from the customer companies can describe their new demands through a web-based request system. The requests are automatically forwarded to the helpdesk service. They conduct an initial filter to select or discard requests. Then, the companies' teams analyse these demands and follow up their status in the tool, from the demand registration until the fulfilment of requirements in a future release.

The case companies face a challenging requirements negotiation due to the interplay with several partners. They must align their own interests and schedules, and negotiate alternative solutions before presenting a proposal to the customer. Negotiation between partners can be either focused on a wider scope of the integration project or it can take place over the set of requirements. According to the project manager of Company A, power conflicts are frequent among partners. Once recognising the business opportunities brought by a specific demand from customers, partners can battle to implement that feature. The requirements negotiation process starts when the project scope is being defined, but it can also happen throughout the integration project. To stay in power, companies frequently attempt to develop features that they do not have sufficient knowledge to implement. It was highlighted by several interviewees that partners who behave in this manner may harm the success of integration project and damage their reputation.

Agreements are easier to achieve with consolidated alliances, when companies already know their specific roles and duties in the network. In new partnerships, there are uncertainties about the partners' true intentions, yielding conflicts among them. According to a project manager from Company B, *"there is a partnership, but each one has his own goals and interests"*. To treat requirements conflicts, meetings are held to discuss goals, benefits and challenges related to each part of the integration scope. During these meetings, companies negotiate what features each partner will implement. Generally, this decision is based on the companies' technical expertise, but dominant partners can also impose their ambition towards a particular set of features that can be strategic for their product evolution. In addition, each company presents an estimative of time, costs and team resources needed to implement the proposed part of the integration scope. After reaching a consensus, partners individually plan their deliveries. The deliverables are described in the 'Vision Document', which consists of the commercial proposal for a particular integration. Then, the project manager and the release manager negotiate specific requirements with the customer.

Requirements analysis follows the negotiation phase. At Company A, it begins with an initial evaluation by the project manager. She analyses the feasibility of a demand received from the ARN document or the web-based request system. At this moment, no estimation technique is used. Once the demand is considered a suitable requirement, it is registered as a 'Customer Change Request' (CRC) in the One Studio tool. Then, the business analyst evaluates the demand to determine if it consists of evolving an existing feature or creating a new

one. Finally, a planning meeting is carried out among the project manager, business and system analysts. At Company B, the business analyst is responsible for performing such analysis. She verifies whether the requirement is sufficiently described and contacts the stakeholder who demanded it to obtain additional information. Finally, the business analyst verifies whether the requirement shall be introduced as part of the standard product or included in a specific version for the customer who requested it.

Requirements are prioritised in the beginning of each Scrum Sprint at Company A. During a meeting between the project manager and the quality team, requirements are classified as billable, strategic or legal. Requirements prioritisation is carried out by the project manager and the business analyst at Company B. They consider criteria such as customer preferences, impact over the product evolution and dependence with other requirements. Their main concern here is to reduce time-to-market. Considering the joint development, system analysts of partner companies promote meetings to discuss integration aspects such as software artefacts that can be reused and integration requirements. Both companies produce a document describing the technical details of the integration project.

The requirements documentation is validated by the product committee at Company A. After being validated by the project manager at Company B, the documentation is forwarded to the product manager, who analyses the alignment of strategic requirements and macro features with the product roadmap. However, the requirements document is not kept up to date reflecting integrated systems evolution. This hampers the construction of a proper historical basis of integration projects. The project manager from Company A claimed that the main reason for not updating the requirements documentation is due to their very short development cycles. Integration projects bring several small changes in requirements because the evolution of functionalities in one product can affect diverse systems. Hence, the companies develop parameterised functionalities to address the impact of changes on a wide range of users. Changes generated by customer needs and legal issues are generally simple and punctual because of the products maturity.

E. Partnerships

Companies A and B have around 10 and 20 partners, respectively. Generally, partnerships are established due to demands from specific clients to integrate products as well as opportunities to enter in new market segments. However, not all product integrations occur with a partner. In these cases, the supplier simply has to integrate the product with other software solutions developed from other companies or with internal software systems from customers. We identified two types of partnerships between the studied companies: existing products integration and joint new product development. The first case usually happens when a company wins a new contract, then they offer to the customer an integrated solution developed with partners. This strategy enables companies to specialise in a particular domain and indicate partners that complement their products. It can also happen in a proactive manner, when the companies conduct their annual roadmap planning and identify strategic market niches they want to enter. Then, the companies propose the development of new products with current partners or search for new partners specialised in those segments. The product manager at Company B highlighted that companies usually start the development of new products with partners based on opportunities identified in previous joint projects. Recently, Companies A and B have developed a new product for the pharmacy retail market together with a third partner.

The evolving relationship among partners requires several actions to align business strategies, such as selection of strategic market niches for all parties, negotiation of pricing models, and agreement on how to conduct joint evolution of products and technologies. After establishing a new alliance resulting from an integration project, companies attempt to eliminate overlaps between products and emphasise complementary features. However, as mentioned by the system analyst of Company B, a common challenge is managing the customisation of several products with the specific needs of customers. Partners need to improve the analysis and understanding of the customers' business processes to facilitate the alignment with products features. We identified that integration projects can follow three strategies (Figure 2):

- **Customer-driven Integration** – customers negotiate separate contracts with each supplier. Customers have strong control over the integration project and suppliers have considerable autonomy to evolve their products independently. In this situation, we did not observe an emerging ecosystem being created;
- **Supplier-driven Integration** – customers are aware of the existence of two or more suppliers. However, customers negotiate the contract directly with one supplier. This supplier assumes the responsibility to control the integration project and to conduct negotiations between customers and suppliers. This scenario enables the creation of an ecosystem among actors, in which suppliers have a shared power towards the evolution of software products;
- **Value Added-Reseller** – one supplier assumes the responsibility for the integration with other suppliers by adding features to its own product. The partnership among suppliers is hidden from the customers. The product commercialisation is dealt directly between the customer and the supplier who owns the main product. In this situation, we observed a growing dominance by the central supplier. The creation of an ecosystem depends on the company's ability to demonstrate the benefits of the partnership to other suppliers.

The supplier-driven integration is the most frequent type of partnership in the studied context. Players establish the sharing of duties and start the negotiation to define who is going to implement what features. This is a very critical phase to decide which supplier will control the most strategic features.

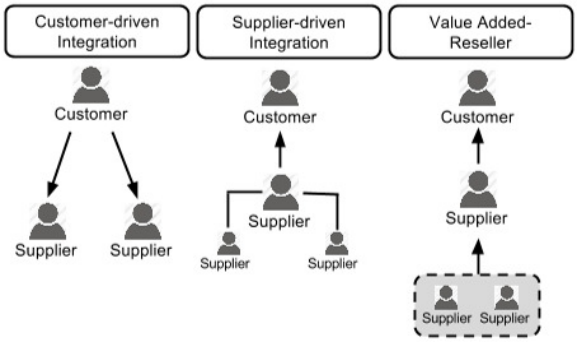

Fig. 2. Partnership strategies.

We observed that technical teams are not explicitly aware of the prosperity of partnerships. Since developers are not involved in strategic decisions, they believe that alliances end after a successful joint integration project, as revealed by one system analyst from Company A: *"this notion of alliance dies once a project is concluded"*. By considering the integration as a temporary effort and not a structural relationship, developers do not understand the long-term opportunities and commitments. In particular, some features may not suit future integrations. Therefore, code quality, evolution, and architectural issues must become structural parts of integration projects with consistent buy-in from the development team.

To address the growing challenges involved in the integration of several products, both companies are developing an integration platform. The companies are negotiating how this platform will be shared and managed by their partners. Top management perceived that the platform is a strategic action to increase the prosperity of their ecosystem. In addition, the release manager and system analyst from Company B are enthusiastic about the benefits of the platform to support the communication among products from different companies. However, other interviewees have different opinions regarding the benefits of the platform. Interviewees from the technical teams of both companies argue that the evolution of the platform is an extra development effort, since the maintenance of the platform is considered quite complex. These divergent viewpoints may indicate the need to better communicate the strategies related to the integration platform. In particular, companies should ensure that decisions about the platform are shared and understood among all levels of the companies.

IV. Discussion

In this section, we present a set of observations to explore the main opportunities and challenges faced by studied companies during the creation of a software ecosystem. These observations are analysed in light of literature in the field.

1. SMEs face a dilemma during the early stages of a software ecosystem: are we competitors or collaborators?

The studied companies confirmed the gains obtained from strengthening relationship with their ecosystem partners. A key benefit obtained from partnerships is the opportunity to enter in new markets that partners are already well established. In addition, new contracts can be obtained by indication from partners who usually propose joint sales to customers. This means that each partner can specialise in their own competencies to provide complementary features within the ecosystem. The availability of integrated products also means that customers will benefit from time and cost reduction.

However, companies within a software ecosystem face an increasing socio-technical complexity, which not only arises from technological challenges to integrate several software systems but also involves battle for power and control of the most valuable product features. Our findings also indicate that managing joint teams brings a number of challenges. For example, technical teams do not fully understand the purpose, direction and responsibilities for each team; managers responsible for a product integration do not have full authority on the partners teams. Another problem we identified is that technical teams are not fully aware of the relevance of some product integrations. Therefore, studied companies should improve internal and external communication channels to highlight the importance of partnerships.

Participants from both companies recognise the positive aspects of collaboration. Nevertheless, they also acknowledge that competition is inevitable due to the fact that they are medium-sized enterprises fighting to survive in the very competitive Brazilian software market. For instance, product roadmaps and other strategic issues are not fully shared among partners. Companies tend to share more information with long-term partners, while new entrants have very little awareness on the products evolution directions. Similar behaviour has been reported in other studies. In a general business perspective, Moore [16] indicates that the complex interplay of competition and collaboration strategies continues during the lifecycle of an ecosystem. Iansiti and Levien [17] propose that participants in a business ecosystem are engaged in mutually dependent relationships that will affect the health of the ecosystem as well as the individual health of their own business. The level of intimacy wanted in a relationship with companies depends on how critical and strategic the product from a partner is [18]. Therefore, partnerships enable sharing of knowledge, technology and increase innovation potential, factors that makes a company an attractive partner [19]. Coopetition is a main driver of innovation and performance. It takes the relationship between the companies to a new level, where players work together to identify innovative requirements and deliver new solutions that address market needs. According to Levy and colleagues [20], to successfully address coopetition, SMEs need to carefully make decisions on what to share, with whom, when, and under what conditions.

2. Roles are volatile in a software ecosystem formed by SMEs

Manikas and Hansen [9] identified that the most common actors in a software ecosystem are keystone, niche player, external developer, independent software vendor and customer. The keystone plays an active and predominant role in the creation and diffusion of value within the ecosystem. Key-

stone companies often own the platform, which gives them a competitive advantage, but also provide resources to other players contributions in a 'win-win' fashion [19].

We did not observe an explicit keystone behaviour in the studied companies. In fact, both companies take the leadership role depending on the market niche in which they are operating. Company B is the most active partner who identifies market opportunities and promotes new alliances. However, Company A has also done so in the past. The shared ownership of the integration platform was considered beneficial by participants to maintain a healthy and balanced partnership. Currently, we did not identify dominators who aim to extract the maximum value of the network without sharing it with other players.

There are several ways to interpret this phenomenon of volatile roles in the ecosystem. The companies are forming a community-oriented ecosystem in which they create flexible committees to promote self-regulation [3]. Other possible explanation is that the ecosystem is not mature enough for a dominator to arise. Another possibility is that an ecosystem composed by SMEs has different dynamics compared to ecosystems created by large organizations who play a central role integrating other players around its own platform. To define a proper direction for the evolution of the ecosystem, the companies need to address several SECO governance issues [21], such as, define clear responsibilities, make business strategy explicit, and decide the level of knowledge sharing. We do believe it is inevitable that the partners will soon observe that their repeated involvement in partnerships requires explicit ecosystem management, as we have analysed in other case studies [22].

3. SMEs participating in a software ecosystem jointly develop and manage a shared platform.

In this scenario of joint development, business and system analysts from both companies participate in several meetings to define integration requirements. These requirements do not refer to feature specification but are rather technical statements to establish the integration process. The impact of integration requirements in all systems must be carefully assessed, since they are part of a shared infrastructure. The teams treat these requirements as reusable assets. By doing that, we can foresee that integration requirements will become platform requirements during the evolution of the SECO. According to Harland and Wust [23], platform requirements involve technical aspects such as software libraries and content databases. The authors claim that platform requirements are the basis of a product platform, which is a development environment used by actors to develop complementary products in the ecosystem.

This discussion of integration requirements is totally aligned with the current efforts of the case companies to build up a central platform that supports their integration projects. Isckia and Lescop [19] describe that a platform enables a composition of functionalities or services that partners can access via a set of common interfaces. The platform will support the development of valuable synergies and complementary innovations for partners and customers. Gawer and Cusumano [24] propose that the development of a successful platform follows four main stages: define the scope of relationships, build the core strategically, build relationships with external complementors, and optimise internal organizational structures. Iansiti and Levien [17] advocate that a platform is especially important for keystone companies to position their leadership and foster their value proposition in the whole ecosystem. In addition, companies use the platform as an instrument to control their influence on the ecosystem [23].

In the studied companies, an initial platform was created and is being maintained by both partners. They are evolving from a *productisation* to a *platformisation* approach [25], fostering a vibrant and potentially larger ecosystem around the platform. In this sense, the companies are embracing mutual dependency that would require closer alignment of their business models. The companies face a hard decision to make: they should share their internal plans with strategic partners, while they do not want to lose their autonomy. On the other hand, they are aware of the importance of the platform to enable future integrations. Hence, the prosperity of their products will depend on the healthy evolution of the platform. Due to resource constraints faced by SMEs, the attraction of third-party developers can be an important strategy to create niche features.

4. Partner companies must synchronise product strategies to sustain ecosystem success.

Nowadays, software companies are expected to provide an overall view of the product evolution and long-term decision making about future product releases [26]. To effectively integrate products from different partners, companies should make an effort to synchronise their product releases. By doing that, partners can better plan and structure their future releases aligned with the product evolutions from other players. This strategy can also support the maintenance of integrated solutions. Although the studied companies are aware of the problems brought by the lack of product release visibility, they are not fully prepared to evolve their systems in a synchronised and jointly fashion. Frequently, the evolution of one product may generate incompatibilities with other solutions that are part of an integration. For example, there may be conflicts related to features functioning or even removal of features due to potential disuse by another integrated system.

To address these mismatches, partners are gradually aligning features prioritisation in future product releases. In particular, both companies follow agile methods such as SCRUM and Kanban for software development and project management. These practices could also be applied in software product management context. Vlaanderen et al. [27] introduce the notion of product management sprint as a cycle performed immediately before the development sprint. In these product management sprints, partners could not only analyse interdependencies between features, but also start to relate their product strategies. We can foresee that this will lead to the convergence of product roadmaps, supported by the extended release cycles currently targeted by the companies. By synchronising product releases and roadmaps, partners can simplify integrations and establish a self-regulation mechanism. The frequent interaction and

feedback between the companies can promote the openness and transparency of strategies. As a consequence, the health of the ecosystem can be improved.

5. A partner's power has a strong influence on requirements negotiation.

Requirements are negotiated by the case companies in a three-step approach. Primarily, partners define their roles in the integration project and divide the scope of systems integration. Then, the companies identify and prioritise a set of requirements with customers. Finally, a third round of negotiation is performed among partners, who suggest new features for each others products and establish integration requirements. According to interviewees, several challenges arise both during the interactions among partners and with partners and customers.

The tendency of customers to require a full customisation of the products can lead to a tough negotiation. Suppliers must deeply understand the effects of customers' demands to agree on scalable adaptations and reduce maintenance costs, since products are developed for a wider market. However, partners' key challenge consists on disputes for a reasonable division of responsibilities and feature implementation. The strategic positioning of the companies in the software ecosystem strongly affects the negotiation of requirements. This is clearly related to the notion of power, which is a common issue affecting decision-making in requirements engineering [28].

In the studied companies, power relationships are perceived in varied moments. Once inviting a partner to provide an integrated solution and thereby sharing its pool of customers, the company has a greater power over the negotiation. Therefore, the company can select the requirements that aggregate more value for the product. This situation represents the legitimate authority of the inviter company, who may act as a value-added reseller and exert a more pervasive influence over the negotiation process. Another illustrative example occurs when the invitee controls the agenda, since his participation is due to a greater experience on a particular niche or technology. In this case, the direction of power relationships is oriented by the skill or knowledge possessed by the partner, configuring an expert power. Finally, one product may act as a central hub and send information to the other systems. The integration project will be centred around this product, increasing the bargaining power of the supplier during the requirements negotiation. Milne and Maiden [28] call this situation as referent power, which here consists of the strategic characteristics of one product that increase the power of the product's supplier.

We observed that the dynamics of requirements negotiation is constantly changing and this can be attributed to the emergent character of the investigated software ecosystem. According to Moore [16], the lifecycle of an ecosystem involves the phases of birth, expansion, leadership and self-renewal. The case companies are currently building up the software ecosystem by attracting external players to provide solutions that satisfy niche markets. As SMEs, partners aim to complement their products and increase their synergy at this birth stage. However, companies have to face battles for strategic requirements that allow them to enter in new market niches and thrive in the ecosystem. Such conflicts of interests are very common in emergent ecosystems, since partners have not clearly established power configuration and coordination mechanisms are incipient or even inexistent. Moreover, the shared responsibility over requirements leads to problems of mutual understanding [29]. To address the positioning issues involved in these negotiations, partners must understand the co-evolution of their products within the ecosystem.

V. Conclusion

In this paper, we reported on a case study of two small to medium market-driven organisations that are forming a software ecosystem. The experiences of the studied companies have indicated that by selling their products as complementary solutions to address the needs of specific customers, the companies can access new niches and reduce time to market. Since SMEs face limited resources, tightening collaboration is an important strategy to their survival. However, companies must balance the inevitable competition among them.

The findings from the case study can be summarised as follows. The roles and relationships between partners in young ecosystems are extremely volatile and flexible. The partnership is maintained as long as there exists strategic alignment, translated in efforts to synchronise product releases and roadmaps. Such relationships lead to relatively fast innovations: as collaboration is mostly pragmatic, integrations are built with a short-term view. To counter such downside of the pragmatic view, software vendors can make integration and partnering more structural through SECO governance issues. Accordingly, partners are constructing a shared platform, which shall facilitate the introduction of new features and support the prosperity of integrated products. We also observed that, even in such young relationships, deliberate displays of power and claiming of specific features during requirements negotiation. Companies must share their internal plans and pools of customers with partners, but they do not want to reduce their autonomy. This illustrates a natural part of the process, which is not harmful as long as both companies maintain their benefits. Finally, this paper can be seen as a call to action for software firms to structurally develop their partnering relationships and communication channels, as collaboration seems to be one of the main ways to gain competitive edge in the software sector.

This case study has an explorative purpose. Therefore, we aim to further investigate the presented observations through new studies with SMEs in Brazil and other countries. We plan to explore how other similar software ecosystems deal with the issues investigated in this paper. In addition, with sufficient case material we also aim to create an ecosystem maturity model, with processes and practices that make a company more mature in terms of ecosystem governance.

Acknowledgment

We are grateful to the participants from the studied companies for sharing their experiences and providing valuable

information to this study. This research has been partially funded by CAPES-Brazil (grant 1833-13-8).

REFERENCES

[1] G. Hanssen and T. Dybå, "Theoretical foundations of software ecosystems," in *Proc. 3rd International Workshop on Software Ecosystems*, 2012, pp. 6–17.

[2] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," presented at the ICSE Companion, 2009, pp. 187–190.

[3] G. Hanssen, "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory," *Journal of Systems and Software*, vol. 85, no. 7, pp. 1455–1466, 2012.

[4] L. Yu, "The market-driven software ecosystem," *IT Professional*, vol. 15, no. 5, pp. 46–50, 2013.

[5] M. Khalil, P. Dominic, H. Kazemian, and U. Habib, "A study to examine if integration of technology acceptance model's (tam) features help in building a hybrid digital business ecosystem framework for small and medium enterprises (smes)," in *Proc. of Frontiers of Information Technology. FIT'11*, 2011, pp. 161–166.

[6] J. Bosch, "From software product lines to software ecosystems," in *Proc. 13th International Software Product Line Conference*, 2009, pp. 111–119.

[7] O. Barbosa, R. Pereira, C. Alves, C. Werner, and S. Jansen, "A systematic mapping study on software ecosystems from a three-dimensional perspective," in *Software Ecosystems - Analyzing and Managing Business Networks in the Software Industry*, S. Jansen, S. Brinkkemper, and M. Cusumano, Eds. Edward Elgar Pub. Ltd., 2013, ch. 3, pp. 59–83.

[8] F. Santana and C. Werner, "Towards the analysis of software projects dependencies: An exploratory visual study of software ecosystems," in *Proc. 5th International Workshop on Software Ecosystems*, 2013, pp. 7–18.

[9] K. Manikas and G. Hansen, "Software ecosystems - a systematic literature review," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1294–1306, 2013.

[10] T. Mens and M. Goeminne, "Analysing the evolution of social aspects of open source software ecosystems," in *Proc. Third International Workshop on Software Ecosystems, CEUR-WS*, 2011, pp. 1–14.

[11] M. Anvaari and S. Jansen, "Evaluating architectural openness in mobile software platforms," in *Proc. Fourth European Conference on Software Architecture: Companion Volume*, 2010, p. 8592.

[12] S. Draxler, A. Jung, and G. Stevens, "Managing software portfolios: A comparative study," in *End-User Development*, M. Costabile, Y. Dittrich, G. Fischer, and A. Piccinno, Eds. Springer Berlin Heidelberg, 2011, pp. 337–342.

[13] P. Runeson and M. Host, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131–164, 2009.

[14] (2013) Weft qda. [Online]. Available: http://www.pressure.to/qda/

[15] (2014) Mps.br. [Online]. Available: http://www.softex.br/mpsbr/

[16] J. Moore, "Predators and prey: a new ecology of competition," *Harvard Business Review*, vol. 3, no. 71, pp. 75–86, 1993.

[17] M. Iansiti and R. Levien, "Strategy as ecology," *Harvard Business Review*, vol. 3, no. 82, pp. 68–78, 2004.

[18] J. van Angeren, V. Blijleven, and S. Jansen, "Relationship intimacy in software ecosystems: A survey of the dutch software industry," in *Proc. International Conference on Management of Emergent Digital EcoSystems*, 2011, pp. 68–75.

[19] T. Isckia and D. Lescop, "Open innovation within business ecosystems: A tale from amazon.com," *Communications & Strategies*, vol. 2, no. 74, pp. 37–54, 2009.

[20] M. Levy, C. Löbbecke, and P. Powell, "Smes co-opetition and knowledge sharing: The is role," in *Proc. 9th European Conference on Information Systems*, 2001, pp. 640–652.

[21] A. Baars and S. Jansen, "A framework for software ecosystem governance," in *Proc. Third International Conference on Software Business*, 2012, pp. 168–180.

[22] S. Jansen, S. Brinkkemper, J. Souer, and L. Luinenburg, "Shades of gray: Opening up a software producing organization with the open software enterprise model," *Journal of Systems and Software*, vol. 85, no. 7, pp. 1495–1510, 2012.

[23] P. E. Harland and S. Wst, "Strategic, brand and platform requirements for an interactive innovation process in business ecosystems," in *Proc. 18th International Conference on Engineering, Technology and Innovation*, 2012, pp. 1–9.

[24] A. Gawer and M. Cusumano, *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, 2002.

[25] P. Artz, I. van de Weerd, S. Brinkkemper, and J. Fieggen, "Productization: Transforming from developing customer-specific software to product software," in *Proc. First International Conference on Software Business*, 2010, pp. 90–102.

[26] T. Suomalainen, O. Saloi, P. Abrahamsson, and J. Simil "Software product roadmapping in a volatile business environment."

[27] K. Vlaanderen, S. Jansen, S. Brinkkemper, and E. Jaspers, "The agile requirements refinery: Applying scrum principles to software product management," *Information & Software Technology*, vol. 53, no. 1, pp. 58–70, 2011.

[28] A. Milne and N. A. M. Maiden, "Power and politics in requirements engineering: Embracing the dark side?" *Requir. Eng.*, vol. 17, no. 2, pp. 83–98, 2012.

[29] S. Fricker and M. Glinz, "Comparison of requirements hand-off, analysis, and negotiation: Case study," in *Proc. 18th IEEE International Conference on Requirements Engineering*, 2010, p. 167.