Adoption of OSS components: a goal-oriented approach

Lidia López[a], Dolors Costal[a*], Claudia P. Ayala[a], Xavier Franch[a], Maria Carmela Annosi[b], Ruediger Glott[c], Kirsten Haaland[c]

[a]*Universitat Politècnica de Catalunya, Ed. Omega, Jordi Girona 1-3, 08034 Barcelona, Spain*
[b]*Ericsson Telecomunicazioni, Via Madonna di Fatima, 2, 84016 Pagani, Salerno, Italy*
[c] *UNU-MERIT, Keizer Karelplein 19, Maastricht, The Netherlands*

Abstract

Open Source Software (OSS) has become a strategic asset for a number of reasons, such as short time-to-market software delivery, reduced development and maintenance costs, and its customization capabilities. Therefore, organizations are increasingly becoming OSS adopters, either as a result of a strategic decision or because it is almost unavoidable nowadays, given the fact that most commercial software also relies at some extent in OSS infrastructure. The way in which organizations adopt OSS affects and shapes their businesses. Therefore, knowing the impact of different OSS adoption strategies in the context of an organization may help improving the processes undertaken inside this organization and ultimately pave the road to strategic moves. In this paper, we propose to model OSS adoption strategies using a goal-oriented notation, in which different actors state their objectives and dependencies on each other. These models describe the consequences of adopting one such strategy or another: which are the strategic and operational goals that are supported, which are the resources that emerge, etc. The models rely on an OSS ontology, built upon a systematic literature review, which comprises the activities and resources that characterise these strategies. Different OSS adoption strategy models arrange these ontology elements in diverse ways. In order to assess which is the OSS adoption strategy that better fits the organization needs, the notion of model coverage is introduced, which allows to measure the degree of concordance among every strategy with the model of the organization by comparing the respective models. The approach is illustrated with an example of application in a big telecommunications company.

Keywords: Open Source Software; OSS adoption; Ontologies; Conceptual Modelling; i-star.

1. Introduction

Open Source Software (OSS) has become a driver for the primary and secondary information technology (IT) sector. Estimates exist that in 2016, as high as 95% of all commercial software packages will include OSS components [1]. Nevertheless, IT companies and organizations still face numerous difficulties and challenges when making the strategic move to the OSS way of working. Like any new technology, OSS is aligned with new challenges, which mainly derive from the way OSS is produced and the culture and values of OSS communities

In fact, OSS adoption impacts far beyond technology, because it requires a change in the organizational culture and reshaping IT decision-makers mindset. The way in which OSS adoption affects and shapes business models [2][3] is becoming object of increasing

*Corresponding author. Tel. +34 934137892

Email addresses: llopez@essi.upc.edu (L.López), dolors@essi.upc.edu (D. Costal), cayala@essi.upc.edu (C.P. Ayala), mariacarmela.annosi@ericsson.com (M.C. Annosi), glott.ruediger@gmail.com (R. Glott), kirstenhaaland@gmail.com (K. Haaland)

Preprint submitted to Data & Knowledge Engineering

attention, and as a result, several OSS business models have been identified so far [4][5][6].

Leveraging OSS adoption strategies with the organization context is a challenging task per se, as it implies reconciling them from very different perspectives [7]. Organizational modelling can provide a way to define the organization's goals and to serve as the context in which processes operate and business is done. However, an important aspect needs to be taken into account: OSS-based solutions are not developed, and do not exist, in isolation. Instead, they exist in the wider context of an organization or a community, in larger OSS-based business ecosystems, which include groups of projects, companies that may be competitors, OSS communities, regulatory bodies, etc. Any approach to organizational modelling for OSS ecosystems needs to consider this collaborative dimension.

In this paper, in order to support organizations that would like to adopt OSS (hereafter, OSS adopter) and analyse the implications of such adoption, we propose the use of goal-oriented models using the *i** approach. Organizations can be described as actors, strategic and operational goals that appear in their rationale decomposition, and collaborative needs can be represented through dependencies. We describe six different OSS adoption strategies in terms of models that can be used as a reference for understanding and assessing the impact of the OSS adoption strategies on the OSS adopter organization, as well as complementing the OSS adopter organizational model.

The remainder of the paper is organized as follows. Section 2 introduces the basic concepts needed in the paper. Section 3 presents the research method followed. Sections 4 to 6 develop the main contributions of the paper: the OSS ontology used, the arrangement of its elements into models for the OSS adoption strategies and the application of such models. Section 7 provides details on how the resulting approach fits to the case of one of the RISCOSS EU-funded project industrial partners (www.riscoss.eu). Last, Section 8 provides conclusions and future work.

2. Background

This section presents a set of strategies regarding the way organizations can adopt OSS. We also introduce the main concepts of the *i** modelling framework used for the OSS adoption strategy models.

2.1. OSS Adoption Strategies

Business models are abstract conceptual models that represent the business and money earning logic of a company in a structured way [8]. There is a vast array of OSS business models and business model types that have been identified by numerous authors. For instance, Chang et al. [9] have distinguished four models that secure sustainability of OSS organizations:

- *community*: the cost of sustaining the product or service is covered by building a community of users and industry partners who agree to cooperate on development work and maintenance
- *subscription*: requires users to pay subscription costs to a company in order to obtain maintenance and support
- *dual license* (*commercial*): the users have the choice between a free version of a software that gets no support and a version that must be paid but comes along with guaranteed support, maintenance and service models or with additional features that are only available as proprietary software
- *central support*: refers to a central body that provides robust releases and support for open source products that are of strategic importance to its community

Daffara [10] has identified six basic OSS business models and one remainder group, largely coinciding with the five business models Dornan [11] has spotted in the OSS market:

- *product specialists*: the code is completely open and revenues are achieved through services related to using, maintaining and adapting the software
- *build (or run) hardware*: this business model aims at making hardware more profitable through installing OSS on commodity components
- *propietary components*: combines proprietary and open-source code, essentially holding back some functionality from what is released for free
- dual licensing: as above
- *badgeware*: reinvention/extension of a previous license constraint
- *platform provision*: companies that provide selection, support, integration and services on a set of projects, collectively forming a tested and verified platform

These and similar classifications of OSS business models rely on the concrete way in which OSS components are adopted in the organization. That is, each organization should define its own OSS adoption goals and determine the actions involved to achieve these goals (i.e., to define the strategy to be followed to fulfil its OSS business model). We have investigated this issue in the FP7 RISCOSS project (www.riscoss.eu) and we have identified some OSS adoption strategies usually followed by the industry. These strategies are described below. They are the ones considered in the rest of the paper:

- *OSS Acquisition* means to use existing OSS code without contributing to the underlying OSS project/community.
- *OSS Integration* means the active participation of an organization in an OSS community with the purpose to share and co-create OSS. In this case, being part of the community in order to benefit from the commonly created OSS components is the key goal of the OSS strategy; it is not necessary for the adopter organization to play a leading role within the community.
- *OSS Initiative* means to initiate an OSS project and to establish a community around it. Usually, the key goal of this strategy is to create community support, but in contrast to the OSS Integration strategy, the adopter establishes the community as a resource that directly serves the company's business strategy and model. As a

consequence, exercising control over the OSS community is typical for this strategy.

- *OSS Takeover* means to take over an existing OSS project/community and to control it. The main difference from the OSS Initiative strategy is that the OSS community already exists.
- *OSS Fork* means to create an own independent version of the software that is available from an existing OSS project or community. This strategy is usually followed when an OSS community on which the adopter organization depends develops in directions that contradict or hamper the organization's business goals. Exercising control over the forked community is not necessary, as the forked community should consist of developers that share the adopter organization's view on how the community and the software should evolve.
- *OSS Release* implies that the organization releases bespoke software as OSS but does not care whether an OSS community takes it up or forms around it. This strategy can, for instance, be observed in the public sector, when software owned by public bodies is released under an OSS license and made available to other public bodies via a repository.

2.2. The i Goal-Oriented Framework*

*i** [12] is a goal and agent oriented framework formulated for representing, modelling and reasoning about socio-technical systems. It offers a good response to our needs of representing expectations of OSS adopters that characterize OSS adoption strategies. Its modelling language (the *i** language) is composed basically of a set of constructs which can be used in two types of models described below.

The Strategic Dependency (SD) model, allows the representation of organizational *Actors*, specialized on *Roles*, *Positions* and *Agents*. Actors can be related by *is-a*, *is-part-of*, *covers*, *instance-of*, *plays* and *occupies* relationships. Actors can also have social dependencies. A *Dependency* is a relationship among two actors: one of them, named *Depender*, depends for the accomplishment of some internal intention on a second actor, named *Dependee*. The dependency is then characterized by an intentional element (*Dependum*) which represents the dependency's element and the *Strengths* associated to each actor, representing the importance of achieving the dependum (at the depender's side) and the difficulty of producing it (at the dependee's side). The primary *Intentional Elements* are: *Resource*, *Task*, *Goal* and *Softgoal*. A softgoal represents a goal that can be partially satisfied, or a goal that requires additional agreement about how it is satisfied.

The Strategic Rationale (SR) model represents the internal actors' rationale. The separation between the external and internal actor's worlds is represented by the actor's *Boundary*. Inside this boundary, the rationality of each actor is represented using the same types of intentional elements described above. Additionally these intentional elements can be interrelated by using one of the following relationships: *Means-end* (e.g., a task can be a mean to achieve a goal), *Contributions* (e.g., some resource could contribute to

reach a quality concern or softgoal) and *Decompositions* (e.g., a task can be divided into subtasks).

3. Research Approach

This research has been performed in the context of the European FP7 RISCOSS project [13], which aims to support OSS adopter organizations to understand, manage and mitigate the risks associated to OSS adoption. The consortium includes 5 industrial partners from public and private sectors, with diverse OSS adoption contexts, which have served to formulate the results presented here (as described below). In line with this objective, this paper focuses on supporting organizations in analysing the implications of adopting a particular OSS adoption strategy. With this aim, our research approach was based on 3 complementary stages corresponding to 3 research questions:

RQ1: How to characterise OSS projects?

This is aimed to understand the relevant practices (mainly in terms of activities and resources) taking place in the context of OSS projects, especially related to software development and community management. We conducted a Systematic Literature Review (SLR) in order to identify existing ontologies on the field. We analysed them with respect to our objectives and complemented the results with knowledge coming from RISCOSS industrial partners. The ontology is described in Section 4.

RQ2: How do OSS elements map to OSS adoption strategies?

In this question, we wanted to inquiry about how the elements emerging from RQ1 map into the different OSS adoption strategies enumerated in Section 2.1. Our aim was mainly to emphasize and represent the different effects that each OSS adoption strategy has over the OSS adopter organization. As a result, the detailed definition of the different adoption models, one for strategy, was obtained. See Section 5 for details.

RQ3: How do OSS adoption strategies relate to the goals of the OSS adopter organization?

OSS adoption strategies' models resulting from RQ2 were mostly focused, as mentioned in the context of RQ1, on software development and community management activities and resources. However, in order to understand the impact of the activities and resources enclosed in each adoption strategy model, we needed to understand their relationship to the OSS adopter's organizational goals. Therefore, we held some off-line workshops with the five RISCOSS industrial partners, and ended up with a set of related goals that were integrated into the models. Furthermore, we have devised some guidelines to support the matching process of the proposed OSS adoption strategies' models with the models of the OSS adopter organization. We have formalized the application of these guidelines to operationalize the usage of the OSS adoption strategies models in the context of any OSS adopter organization.

It is important to highlight that these three RQs and their corresponding stages have a formative character as they were aimed to conceive the OSS adoption strategies models. In all the stages, the industrial partners of the RISCOSS project have been involved to shape and endorse our approach in their respective contexts. In this paper, as a proof-of-concept, we provide details on how the resulting approach fits to the case of Ericsson Telecomunicazioni, Italy (TEI), one of the industrial RISCOSS partners.

For the summative evaluation of the approach, we plan to apply the OSS adoption strategies and the corresponding guidelines in other industrial organizations besides the RISCOSS partners, which would become the last stage of our method (see Figure 1).

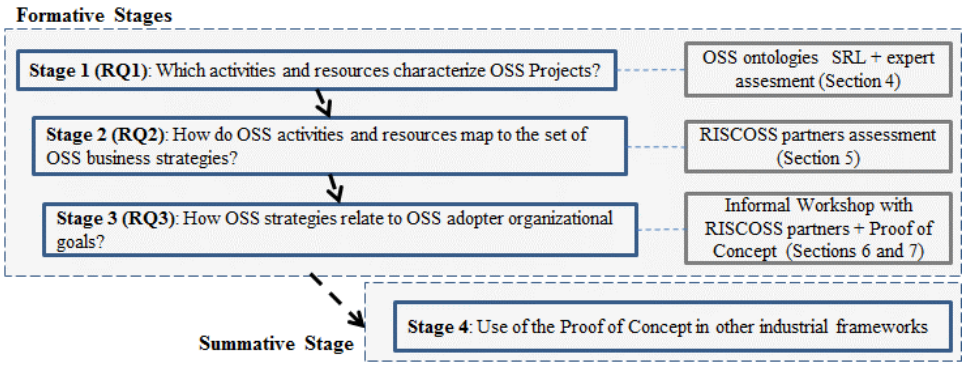

Figure 1: Research method followed in this paper

4. An Ontology for OSS Adoption Strategies

In order to state the basis for defining models that represent the different OSS adoption strategies, we have developed an ontology that embraces terms related to OSS projects and the adoption strategies involved in them. To do so, as a first step, we conducted an SLR using the guidelines defined by Kitchenham [14] with the purpose of identifying existing OSS ontologies. The details of the SLR and the subsequent analysis can be found in [15].

Although the SLR was conducted in order to find ontologies related to OSS field, the search string included some terms besides the term *ontology* (*metamodel*, *glossary*, *taxonomy* and *reference model*) in order to avoid missing papers with a kind of implicit ontology. This also resulted in a large set of initial papers that was considerably pruned: from an initial set of 1214 primary studies, we selected 9 papers to be analysed as papers that contain an ontology (explicit or not). From them, 3 relevant ontologies emerged: Dhruv's ontology [16], OSDO [17][18] and OFLOSSC [19].

We finally chose OFLOSSC as starting point because: (1) it was the most complete amongst the ontologies reviewed (it actually includes parts of the other two), (2) it is an ontology for supporting OSS development communities and covers concepts related to community interactions for developing software. However, it lacks of adoption concepts related to OSS adopter organizations, therefore, we extended it with these missing (with respect to our purposes) concepts. To do so, we performed a thorough analysis of the elements that an adopter organization should consider when participating in an OSS

project by running off-line workshops (i.e., discussions that were centralized in a wiki tool) with partners of the RISCOSS project.

The results of this work have been reported as part of the RISCOSS ontology in [15]. They consist in a complete set of concepts and relations related to the actors, roles, activities and resources that correspond both to OSS communities and OSS adopter organizations. In this paper, we focus on the subset of these concepts related to activities and resources that can be relevant for the interaction that an OSS adopter has with an OSS community according to its adoption strategy. Table 1 lists them. For each element, the table includes its identifier and a brief description. The elements identified have been classified into five groups: software development activities, community-oriented activities, communication activities, personnel activities and resources. We must note that the OSS adopter, when using the OSS component, gets involved in the OSS community to a certain extent that depends on its adoption strategy, and may develop activities of the OSS community (e.g. Act-PATCH). Therefore these activities have been included in Table 1.

Table 1. Activities and resources for OSS adoption strategies

Identifier	Description
Software Development Activities	
Act-SEL	Selection of an OSS component for its deployment or integration in an organization
Act-DEP	Deployment of an OSS component for its actual use in the organization
Act-DEV	Development of an OSS component (specification, design, code)
Act-INT	Integration of an OSS component into another software artifact
Act-TEST-Comp	Testing of an OSS component
Act-TEST-Prod	Testing of a software artifact that integrates an OSS component
Act-MAINT-Comp	Maintenance of an OSS component
Act-MAINT-Prod	Maintenance of a software artifact that integrates an OSS component
Act-PATCH	Development of a patch to correct some bug or add some new feature for an OSS component
Community-Oriented Activities	
Act-NewCOMM	Creation of an OSS community
Act-DECIDE-Roadmap	Decision of the roadmap of an OSS component. It includes planning of releases and which features are included
Act-DECIDE-Acc	Acceptance of a contributor in an OSS community
Act-DECIDE-Wishlist	Decision of the desired features for the next releases of an OSS component (but without a concrete planning)
Act-RELEASE	Making available a software component under OSS license (either first time or an evolution)
Communication Activities	
Act-RepPATCH	Communication of a patch for an OSS component
Act-RepBUG	Report of a bug
Act-SUPP	Any kind of support given to the OSS community (except bug reports and patches; e.g. organising or endorsing sponsoring events)

Act-ASK	Ask about an issue to the OSS community
Personnel Activities	
Act-ACQ-Tech	Acquisition of the necessary knowledge about an OSS component to be able to master its technology
Act-ACQ-Man	Acquisition of the necessary knowledge about managing an OSS community
Act-LEARN	Acquisition of the necessary knowledge about an OSS component to be able to operate it (as end user)
Resources	
Res-OSS-Comp	An OSS component as a software artifact
Res-Tech-DOCUM	Technical documentation of an OSS component
Res-User-DOCUM	User documentation (e.g., tutorials) of an OSS component
Res-PATCH	Patch provided for an OSS component
Res-BUG	Report of a bug or post, etc., referred to an OSS component
Res-NEWFEATURE	Report of desired feature(s) for an OSS component
Res-ROADMAP	Strategy for new features and releases of an OSS component

The concepts in Table 1 exhibit some relationships that can be expressed as ontology properties. They are sub-concepts of *activity* and *resource*, related by two properties; activities *produce* resources and activities *use* resources. Activities themselves are also related by one property: activities *precede* activities. These properties have the sub-properties included in Table 2 relating different sub-concepts of activity and resource.

Table 2. Relationships for OSS adoption strategies activities and resources

Concept ID	Concept ID
Activity *produces* Resource	
Act-RELEASE	Res-OSS-Comp
Act-RELEASE	Res-Tech-DOCUM
Act-RELEASE	Res-User-DOCUM
Act-RepPATCH	Res-PATCH
Act-RepBUG	Res-BUG
Act-DECIDE-Wishlist	Res-NEWFEATURE
Act-DECIDE-Roadmap	Res-ROADMAP
Activity *uses* Resource	
Act-SEL	Res-OSS-Comp
Act-DEP	Res-OSS Comp
Act-INT	Res-OSS-Comp
Act-TEST-Comp	Res-OSS-Comp
Act-MAINT-Comp	Res-OSS-Comp
Act-PATCH	Res-NEWFEATURE
Act-DECIDE-Roadmap	Res-NEWFEATURE
Act-ACQ-Tech	Res-Tech-DOCUM
Act-LEARN	Res-User-DOCUM
Activity *precedes* Activity	

Act-SEL	Act-DEP
Act-SEL	Act-INT
Act-DEV	Act-TEST-Comp
Act-INT	Act-TEST-Prod
Act-NewCOMM	Act-DECIDE-Roadmap
Act-NewCOMM	Act-DECIDE-Acc

5. OSS Adoption Strategies Models

We present next a catalogue of models that characterize each of the OSS adoption strategies described in Section 2. These models are built on top of the ontology presented in Section 4 and each of them combines the ontology elements as required by its corresponding strategy. Models focus on the adopter organization and refer to the particular OSS component under adoption. We have used the *i** framework as modelling approach. The reason is that *i** is an intentional actor-oriented modelling and analysis framework (see Section 2.2), which supports representing and analysing synergistic and conflicting stakeholder interests and decision-making within and across organizational settings.

The models have been discussed in the context of the RISCOSS project running off-line workshops (i.e., discussions that were centralized in a wiki tool) with the project partners.

The main actors involved in the OSS adoption strategies are: the organization that adopts the OSS component (OSS Adopter) and the OSS community that produces it. The six strategies (see Section 2.1) can be characterised depending on: (1) whether the main goal is consuming OSS or producing it (i.e. providing the initial version of the code) and (2) the degree of involvement of the OSS adopter in the OSS community (see Table 3). Regarding the involvement, we have defined three degrees: no involvement, active involvement in the community (i.e. investing some resources for the evolution of the OSS component) and leading the community (i.e. investing some resources in order to try to control the evolution of the OSS component).

Table 3. OSS adoption strategies

	Not involvement	**Active involvement**	**Leading**
Consuming OSS	OSS Acquisition	OSS Integration	OSS Takeover
Producing OSS	OSS Release	OSS Fork	OSS Initiative

The activities performed and the resources produced by each of these actors vary significantly depending on the adoption strategy, and this is the basis of the model construction. For each adoption strategy, we have allocated the activities and resources, presented in Section 4, to the two actors, depending on which one is responsible. Table 4 provides the allocation of activities and resources to the adopter organization actor depending on the adoption strategy followed. The expertise obtained from the RISCOSS

partners has provided us the rationale to choose the most adequate allocation according to the main features of each identified OSS adoption strategy.

Table 4: Adopter activities and resources (rows) per OSS adoption strategies (columns)

	Integration	Initiative	Takeover	Fork	Acquisition	Release
Software Development Activities						
Act-SEL	X		X	X	X	
Act-DEP	X	X	X	X	X	X
Act-DEV		X				X
Act-INT	X	X	X	X	X	X
Act-TEST-Comp						
Act-TEST-Prod	X	X	X	X	X	
Act-MAINT-Comp						
Act-MAINT-Prod	X	X	X	X	X	X
Act-PATCH	X	X	X	X		
Community-oriented Activities						
Act-NewCOMM		X		X		
Act-DECIDE-Roadmap		X	X			
Act-DECIDE-Acc		X	X			
Act-DECIDE-Wishlist	X			X		
Act-RELEASE		X				X
Communication Activities						
Act-RepPATCH	X	X	X	X		
Act-RepBUG	X	X	X	X		
Act-SUPP	X	X	X	X		
Act-ASK	X	X	X	X	X	
Personnel Activities						
Act-ACQ-Tech	X	X	X	X	X	X
Act-ACQ-Man	X	X	X	X		
Act-LEARN	X		X	X	X	
Resources						
Res-OSS-Comp		X				X
Res-Tech-DOCUM		X				X
Res-User-DOCUM		X				X
Res-PATCH	X	X	X	X		
Res-BUG	X	X	X	X		
Res-NEWFEATURE	X			X		
Res-ROADMAP		X	X			

Some general observations on Table 4 are the following:

1. For the strategies that do not require community involvement (acquisition and release), the activities allocated to them are mainly internal-oriented software development activities and not community-oriented or communication activities (except for *Act-RELEASE* in the release strategy case and *Act-ASK* in the acquisition strategy).

2. For the rest of strategies: the organization participates in communication activities (e.g. *Act-RepBUG*) and contributes with their corresponding resources (e.g. *Res-BUG*). Additionally, the organization develops different community-oriented activities depending, mainly, on whether it is exercising control over the community or not. Remarkably, in the initiative and takeover case, the organization decides the community roadmap (*Act-DECIDE-Roadmap*, *Res-ROADMAP*).
3. Two of the strategies, namely, initiative and fork, require that the organization sets up an OSS community (*Act-NewCOMM*).

The activities of maintaining and testing the OSS component (*Act-MAINT-Comp* and *Act-TEST-Comp*) are not allocated to the organization in any strategy meaning that they are basically developed by the OSS community. Taking as a basis the allocation described in Table 2, we have built an *i** model for each strategy. These models have been complemented with two kind of goals: some goals used to structure the model (e.g. *Technical Quality* to embrace the tasks related to acquire skills for using the component *Act-ACQ-Tech* and *Act-Learn*) and some high-level goals and softgoals more related to the strategic goals (e.g. *Take benefit from OSS community*).

In the following subsections we detail the models of each of the identified OSS adoption strategies. All of them have a central task representing the type of OSS adoption strategy (*Acquire OSS Component*, *Integrate OSS Community*, etc.). Considering these tasks as a central element, the models include a set of high-level goals directly attained by the strategy (goals above the task, e.g. *OSS involvement minimized* in Figure 2) and the low-level tasks or resources which are requirements for an adequate application of the OSS adoption strategy (tasks or resources below the central task, e.g. *Act-SEL* and *Act-LEARN* in Figure 2).

5.1. OSS Acquisition

The model for the OSS acquisition adoption strategy (see Figure 2) shows how the adopter organization only obtains the component from the OSS community and does not give back any return to the community. Therefore it can be observed from the model that only outgoing dependencies stem from the adopter organization actor. The adopter is interested on using an existing OSS component (as represented by the goal *Take benefit from OSS community*) but it is not necessarily interested on the successive releases of the adopted OSS component produced by the community for the maintenance of its product (goal *Do not care OSS evolution for maintenance*). For this purpose it does not need to be involved in the OSS community that produces that OSS component (softgoal *OSS involvement minimised*).

The central task *Acquire OSS Component* is composed primarily by all the activities that the adopter performs to select and use the OSS component and test and maintain the software product where the component is used. The adopter depends on the community to obtain the OSS component and its documentation as shown by the dependencies.

Additionally, good technical skills are required as represented by the softgoal *Technical Quality*. Activities *Act-ACQ-Tech* and *Act-LEARN* contribute to achieve these skills.

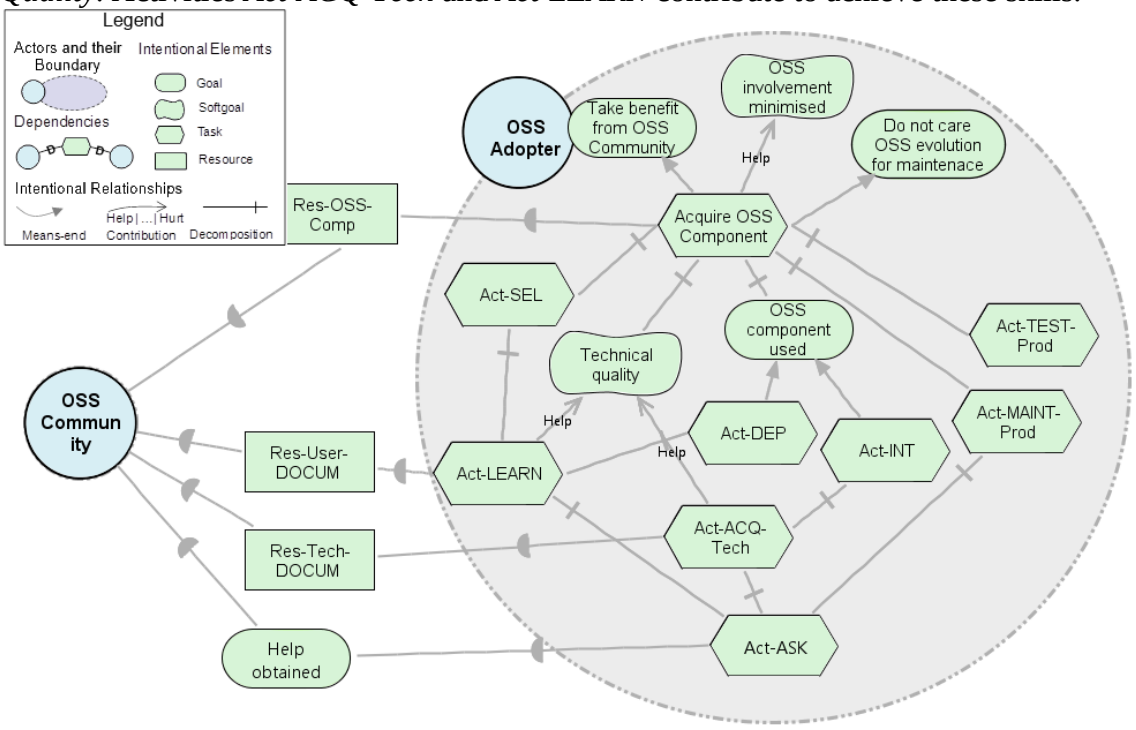

Figure 2: OSS Acquisition model

5.2. OSS Integration

The OSS integration adoption strategy requires sharing and co-creation of OSS between the adopter and the OSS community (see Figure 3). This collaboration means a give-and-take to the benefit of all involved parties. For instance, the adopter might use existing OSS components that have been developed by the community, and pay back in terms of bug reports, patches, etc. Thus, the adopter organization wants to benefit from co-creation (softgoal *Benefit from co-creation taken*); it is interested on being involved in the OSS community (softgoal *OSS involvement*) and on influencing it (softgoal *OSS evolution influenced*).

The task *Integrate OSS Component* is composed by the activities already required by the acquisition strategy complemented with additional activities that the adopter develops as part of its co-creation and collaboration with the community. These activities, grouped under the goal *OSS Community Contributed*, may be reporting bugs, developing and reporting patches or other forms of support such as sponsoring events. Dependencies stem out of these activities reflecting the collaboration with the community. For an advantageous co-creation, the contribution must be aligned to the OSS community culture, as represented by the softgoal *According to OSS Community Practices*. The activity *Act-ACQ-Man* consisting on acquiring knowledge about the community

contributes to it.

A feature of the OSS integration adoption strategy is that the adopter organization relies partly on the successive releases of the OSS component for the maintenance of its product. Thus, the adopter organization might want to influence the community roadmap to ensure that successive releases of the OSS component follow its desired features. The softgoal *OSS Comp Evolves towards Desired Features* under the maintenance activity shows this dependency. The adopter organization defines its expectations for the evolution of the OSS component (*Act-DECIDE-Wishlist*). The community may eventually take this wishlist to decide its roadmap as represented by the dependency with dependum *Res-NEWFEATURE*. The maintenance activity of the company also depends on the *Quality of the Evolved OSS Component* provided by the OSS community.

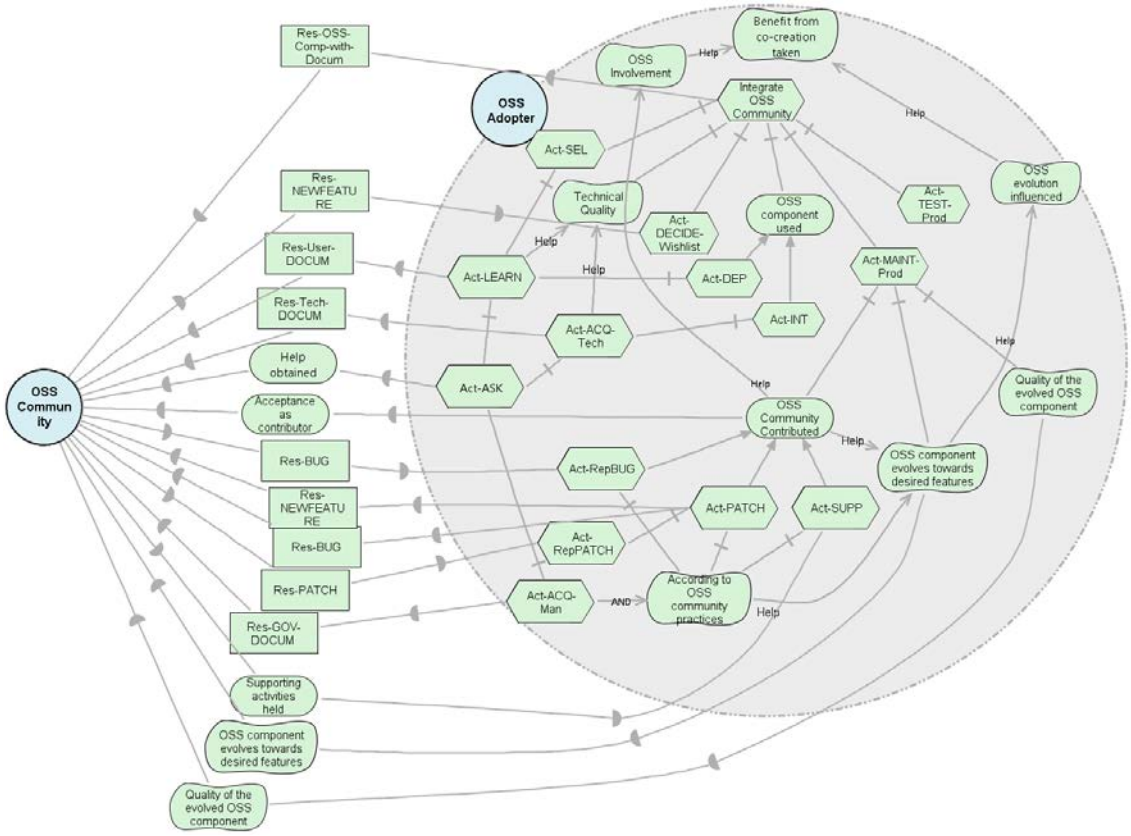

Figure 3: OSS Integration Model

5.3. OSS Initiative

The OSS initiative adoption strategy requires to initiate a new OSS project and to establish a community around it (see Figure 4). In this case, the adopter organization is interested in providing OSS solutions (goal *OSS solutions provided*), controlling its evolution (goal *Steer OSS component evolution*) and, as in the previous integration

strategy, it also considers taking benefit from co-creation (softgoal *Benefit from co-creation taken*) and being involved in the OSS community (softgoal *OSS involvement*).

The central task *Perform OSS Initiative* includes the activities already required by the integration strategy complemented with new ones for building the initial OSS component, releasing it, establishing the OSS community around it and managing the community. Community management skills are required both to create and manage the community as represented by the task *Act-ACQ-Man*. To manage the community implies to take decisions such as those related to the roadmap and the acceptance of contributors (*Act-DECIDE-Roadmap* and *Act-DECIDE-Acc*). For this management to be effective it is required that the community accepts it, as represented by the dependency with dependum *Company Management Accepted*).

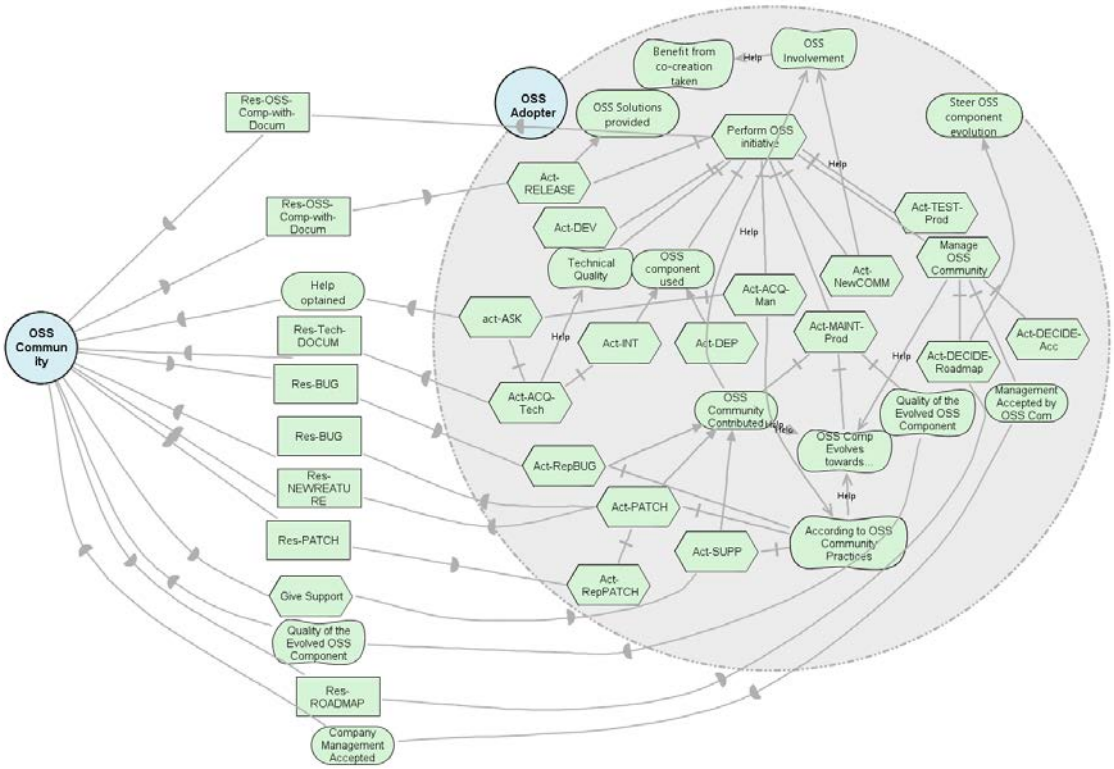

Figure 4: OSS Initiative Model

5.4. OSS Takeover

The OSS takeover adoption strategy means to take over an existing project/community and to control and steer its development (Figure 5). Hence this strategy is similar to the initiative strategy with the difference that, in the takeover case, the OSS community already existed and it is not created by the organization. Consequently, the takeover model is similar to the initiative one (see Section 5.3). The main differences are that the activities related to building the initial OSS component, releasing it, establishing the OSS

community around the component are not included in the takeover case. Conversely, it is required to acquire knowledge about the community in order to have the skills to control it (dependency with dependum *Knowledge acquisition about OSS community*).

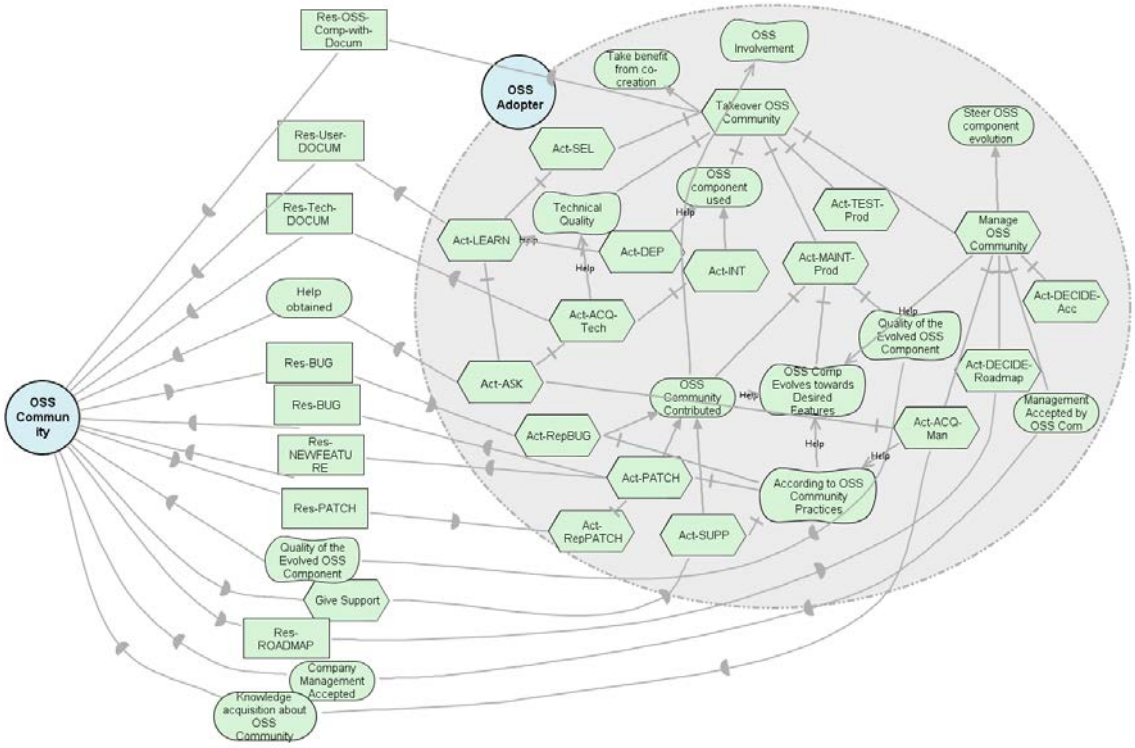

Figure 5: OSS Takeover Model

5.5. OSS Fork

The OSS fork adoption strategy means to create an own independent version of the software that is available from an OSS project or community (see Figure 6). This strategy implies that the organization creates a new community and it does not necessarily exercise control over it and behaves as in the integration strategy.

An organization may be willing to fork an existing OSS project because the evolution of the OSS component does not fit with its needs (goal *Independent OSS component version got*). In this case, the adopter organization does not necessarily want to control the OSS component evolution, but it wants to perform some influence (softgoal *OSS evolution influenced*) in order to take benefit from co-creation (softgoal *Benefit from co-creation taken*) and it wants to get involved in the new community (softgoal *OSS involvement*).

The model of the OSS fork adoption strategy includes two OSS community actors: one corresponding to the existing community that provides the initial software called *Old OSS Community* and the newly created community named *New OSS Community*. Since *i** does

not allow expressing temporal relationships, this sequencing cannot be modelled.

The central task *Perform OSS Fork* is composed primarily by all the activities that the organization performs to use the OSS component and test and maintain the software product where it is used, jointly to the selection of the component to be forked and the creation of the new community. Additionally, it includes the same activities as the integration strategy developed as part of its co-creation and collaboration with the new community such as reporting bugs, developing and reporting patches or other forms of support such as sponsoring events.

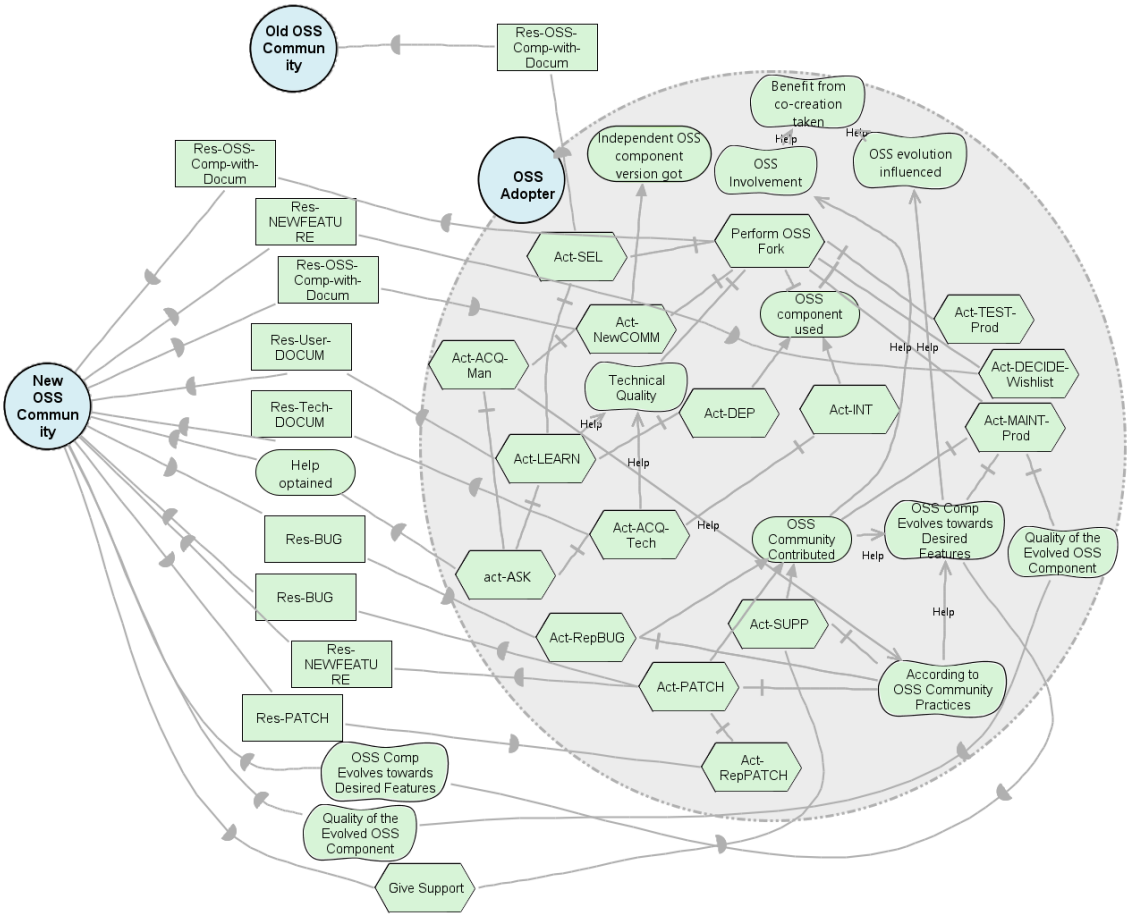

Figure 6: OSS Fork Model

5.6. OSS Release

The OSS release adoption strategy implies that the organization releases bespoke software as OSS but does not care about whether or not a community evolves around it or an existing community makes use of it or develops it further (see Figure 7). Therefore, in this case there is no community involved in the model. The rationale behind this strategy may be to affect an emerging market by releasing software as OSS in order to

control potential competitors. In this strategy, the organization wants to release some product under an OSS license (goal *OSS solutions provided*) without any interest on the OSS community that can be created around it (goal *No OSS involvement*).

The central task *Perform OSS Release* is composed primarily by all the activities that the organization performs to develop, use, test and maintain its software, jointly to the activity of releasing it as OSS component.

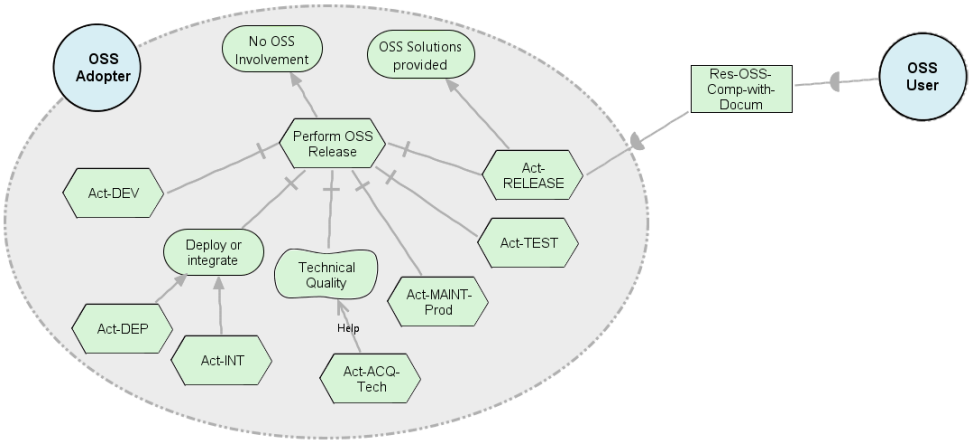

Figure 7: OSS Release Model

6. Guidelines for Applying OSS Adoption Strategy Models in an Organization

OSS adoption strategy models have been developed as generic models; therefore, a question to be answered is how to apply them in a specific situation (in our case, an OSS adopter organization). The question is twofold: first, when is it adequate to apply an OSS adoption strategy model (which responds to the high-level question: which strategy is the most adequate for an organization that wants to go OSS); and second, how to align the organizational model with the strategy model (which responds to the high-level question: what is the effect for the organization to adopt this particular strategy). To answer these questions, we start by assuming the existence of an organizational model that declares the higher-level goals pursued by the organization and using this existing organizational model we select the OSS adoption strategy that is most suited to the organization needs. To answer the second question, we describe the process of extending the model with the elements from the selected OSS adoption strategy model and making the necessary adjustments to these new elements.

6.1. Selecting the OSS Adoption Strategy

Figure 8 shows an excerpt of the organizational model for the fictitious organization *ACME*, that produces the Road Runner Locator (*RR Locator*) product for its customers. The company is interested in reducing in-house development costs, therefore they have

decided to reduce the development effort integrating an OSS component as part of their software, but they are not interested in being involved with the OSS community behind this component.

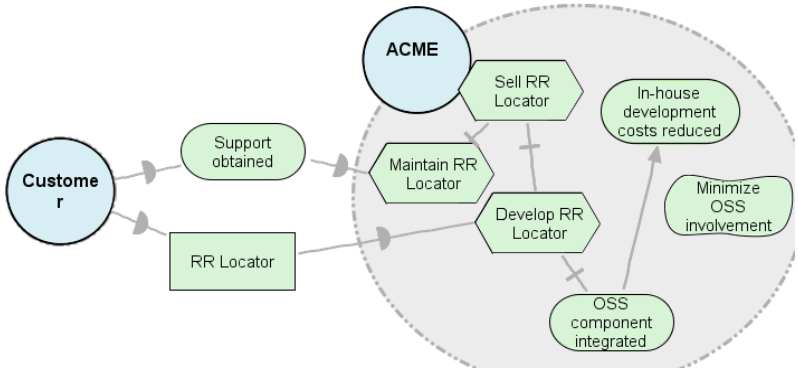

Figure 8: ACME Organizational Model

In order to facilitate the process of finding the more suitable OSS adoption strategy, we have adapted the model matching notion presented by Franch in [20]. In that paper, *i** models are used to describe market segments and software packages in order to evaluate the matching between both models with an *i** organizational model, and select the best software package for the organization needs. In the current proposal, we use the same idea and evaluate the matching between an organizational model and the set of OSS adoption strategy models in order to identify the strategies that better match the organizational goals and eliminate those that clearly do not apply to the specific organization. Franch uses a concept called coverage for classifying the matching results that we adapt for our purposes [20].

The notion of goal matching has also been used in [21][22]. Rolland [21] uses goal matching between organization intentions and the goal/requirements that can be achieved by COTS components, with the purpose of selecting and assembling them. Soffer et al. [22] propose to match enterprise goals, operationalized as business rules, with ERP system capabilities in order to align ERP systems with enterprise requirements.

In the following we first describe the intuitive notion of coverage and then we provide its formalization.

6.1.1. Notion of coverage and its use for selecting strategies

The notion of coverage in the context of selecting an OSS adoption strategy is twofold:

- *Coverage of the organizational model*: it measures to which extent the intentional elements (IEs) that appear inside the Organization actor in the organizational SR model are covered by an OSS adoption strategy model. In other words, it measures to which extent an OSS adoption strategy supports the strategic goals of the organization. This coverage is specified as a percentage (i.e. which percentage of elements is covered).

- *Coverage of the OSS adoption strategy model requirements*: It measures to which extent the tasks and resources, representing *requirements*, which are part of the Organization actor SR model of a specific OSS adoption strategy, are covered by the organizational SR model. In other words, it measures to which extent an adoption strategy can effectively be adopted by an organization from the degree to which the organization fulfils the needs of the strategy. This coverage is specified as a percentage (i.e. which percentage of requirements is covered). The adoption strategy requirements are all the tasks and resources that belong to the decomposition of the strategy central task (this central task was introduced at Section 5).

The aim is finding the OSS adoption strategies that cover as much as possible the organization strategic goals, taking into account that the company has, or is willing to have, the needs required by the strategy. Therefore, any combination where the coverage of the OSS adoption strategy model is incomplete (not a 100% coverage) because some OSS adoption strategy requirements are missing, requires the organization to extend its organizational model in order to apply the strategy. In these situations, the organization should evaluate whether the strategic goals achieved thanks to the strategy are worth the effort needed to fulfil its requirements.

Central to model coverage is the definition of IE coverage. To introduce this notion, we need to refer to the concept of satisfaction of an IE considered as a logical predicate *sat* [23]. The satisfaction of an IE depends on its type: goal satisfaction means that the goal attains its desired state; task satisfaction means that the task follows its defined procedure; resource satisfaction means that the resource is produced or delivered; softgoal satisfaction means that the modelled condition fulfils some agreed fit criterion. In the case of softgoals, the predicate is not indicating if the softgoal is satisfied or not, it is indicating if it is satisfied enough (satisficed). We use the word satisfaction for all kinds of IEs for simplicity. The modeller should identify the pairs of equivalent elements in terms of satisfaction (CC1), and for the others identify whether the element in one model is satisfied by one in the other model (CC2) or the other way round (CC3). Then:

Definition 1. *Intentional element coverage*

Given an IE ie_A belonging to an SR model A and given another SR model B, we define the notion of *intentional element* coverage (by an SR model), *is_covered*(ie_A, B), according to the following four cases:

- **CC1.** There is an intentional element $ie_B \in B$ that is equivalent, in terms of satisfaction, to ie_A, sat(ie_A) $\Leftrightarrow$ sat(ie_B). In this case, *is_covered*(ie_A, B) = true.
- **CC2.** There is an intentional element $ie_B \in B$ whose satisfaction ensures that of ie_A, sat(ie_B) $\Rightarrow$ sat(ie_A). In this case, *is_covered*(ie_A, B) = true.
- **CC3.** There is an intentional element $ie_B \in B$ whose satisfaction is part of that of ie_A, sat(ie_A) $\Rightarrow$ sat(ie_B). However, since this does not guarantee that ie_A is satisfied, we consider that in this case, *is_covered*(ie_A, B) = false.

CC4. There are no intentional elements in *B* for which an implication relationship with ie_A can be found. In this case, *is_covered*(ie_A, *B*) = false.

Note that this definition can be extended into subsets of IEs in a straightforward manner.

As an example, Table 5 shows the coverage of the ACME organizational model (Figure 8) by the OSS Acquisition strategy model (Figure 2) (organizational model coverage). The coverage of the ACME organizational model by the strategy is acceptable (66,6% of elements covered). Furthermore, the only element not related at all with the strategy is the task *Sell RR locator* which, nevertheless, represents a kind of activity not addressed by the OSS adoption strategies. In addition, *Develop RR Locator* even if not total, has some kind of partial coverage which could be considered as a mitigation to the satisfaction of this element (this kind of argumentation would be part of the analysis of results, to be made during the process of deciding the most appropriate OSS adoption strategy for the organization). Therefore, the OSS Acquisition strategy seems a good choice to be applied to the organization.

Table 5: Organizational model coverage

OSS Organizational model	Coverage case	Coverage by OSS adoption strategy model (strategy IEs appear underlined)
Sell RR Locator	CC4	Missing
Maintain RR Locator	CC1	Act-Maint-Prod ⇔ Maintain RR Locator
Develop RR Locator	CC3	Develop RR Locator ⇒ Act-INT Develop RR Locator ⇒ Act-TEST-Prod
OSS component integrated	CC2	Acquire OSS component ⇒ OSS component integrated
In-house development cost reduced	CC2	Take benefit from OSS Community ⇒ In-house development cost reduced
Minimize OSS involvement	CC1	OSS involvement minimized ⇔ Minimize OSS involvement

Table 6 shows the coverage of the OSS acquisition adoption strategy requirements by the ACME organizational model. The requirements are all the tasks and resources that decompose the central task *Acquire OSS component* in the Acquisition model (Figure 2). In this case, the coverage is not optimal (50% of coverage). The requirements related to achieve the technical skills in order to use the component (*Act-ACQ-Tech*, *Act-LEARN* and *Act-ASK*) are missing. Therefore, the organizational model should be extended by including them, if ACME decides to apply the strategy. Note that *ACT-DEP* is not so fundamental to be covered, since there is another means to achieve the goal *OSS Component Used* (*Act-INT*, see Figure 2) which is covered by the organizational model allowing us to discard this requirement.

Table 6: OSS acquisition strategy requirements coverage

OSS Acquisition strategy	Coverage	Coverage by Organizational Model
Act-SEL	CC2	OSS component integrated ⇒ Act-SEL
Act-ACQ-Tech	CC4	Missing
Act-LEARN	CC4	Missing
Act-ASK	CC4	Missing
Act-DEP	CC4	Missing
Act-INT	CC1	OSS component integrated ⇔ Act-INT
Act-TEST-Prod	CC2	Develop RR Locator ⇒ Act-TEST-Prod
Act-MAINT-Prod	CC1	Maintain RR Locator ⇔ Act-MAINT-Prod

6.1.2. Formalization of the *i** framework

We present next an algebraic formalization of *i** based on the definition provided in [23]. The general layout consists on defining elements as tuples of sub-elements and then functions with a meaningful name to obtain these sub-elements. In a nutshell, an *i** model (*M*) contains actors (*A*), dependencies (*DL*), dependums (*DP*) and actor links (*AL*). Actors contain intentional elements (*IEs*, of type: *goal, softgoal, task* and *resource*) connected by IE links (*IEL*) of different types (*means-end, task-decomposition* and *contribution*). Dependencies connect IEs inside actors (although the IEs are not shown when the actors remain closed) and have a dependum (that is also an *IE*). A model element *n* is identified by its name, *name(n)*. Other auxiliary functions with intuitive meaning are used in the formalization; for instance, the function *actors(M)* returns the set of actors of a model, *actors(M) = A*. Table 7 summarizes the formal definitions used in this section.

Table 7. Formal definition of the *i** language as used in this paper.

Concept	Definition	Components
*i** model	$M = (A, DL, DP, AL)$	*A*: set of actors; *DL*: set of dependencies *DP*: set of dependums; *AL*: set of actor specialization links
Actor	$a = (n, IE, IEL)$	*n*: name; *IE*: set of IEs; *IEL*: set of IE links
IE	$ie = (n, t)$	*n*: name; *t*: type of IE, $t \in \{goal, softgoal, task, resource\}$
IE link	$l = (x, y, t, v)$	*x, y*: IEs (source and target) *t*: type of IE link, $t \in IET$ $IET = \{means\text{-}end, task\text{-}decomposition, contribution\}$ with $target(means\text{-}end) \neq softgoal$ $target(task\text{-}decomposition) = task$ $target(contribution) = softgoal$ *v*: contribution value, $v \in CLV$ with $CLV = CT+ \cup CT- \cup \{Unknown\}$ $CT+ = \{Make, Some+, Help\}$,

		CT− = {Break, Some-, Hurt}
Depen-dency	*d = ((dr, ie$_r$, s$_r$), (de, ie$_e$, s$_e$), dm)*	*dr, de*: actors (depender and dependee respectively) *ie$_r$, ie$_e$, dm:* IEs (depender, dependee and dependum, respectively), *ie$_r$ ∈ IE(dr), ie$_e$ ∈ IE(de), dm ∈ DP* *s$_r$, s$_e$*: strengths, *s$_r$, s$_e$∈{open, committed, critical}* *dm*: (n, t) (see IE) *actor(dr) ≠ actor(de)* (an actor cannot depend on itself)
Actor link	*al=(a, b, t)*	*a, b: actors; t = type of actor link, t∈ {is-a, is-part-of, plays, occupies, covers}*
Derived concepts		
intentionalElements(IEL)		*{x | (x, ie, t, v) ∈ IEL ∨ (ie, x, t, v) ∈ IEL}*
descendants(ie, IEL)		*{x | (x, ie, t, v) ∈ IEL ∨* *(∃y: (y, ie, t, v) ∈ IEL ∧ x ∈ descendants(y, IEL))}*

6.1.3. Formalization of the notion of coverage

In this section we formalize the two notions of coverage introduced in Section 6.1.1. Since they are similar, both of them rely in an auxiliary definition of coverage of a set of IEs, which in its turn is defined on top of the intentional element coverage introduced also in Section 6.1.1 (*is_covered*). To illustrate the different concepts, we use the example presented in Figure 9. It represents two sets of intentional elements *A* and *B* such that a_1 and b_1 are equivalent, a_4 is covered by b_5 (but are not equivalent), and b_2 and b_3 are covered by a_3 (without being equivalent either). We assume that neither a_3 nor b_4 is not covered.

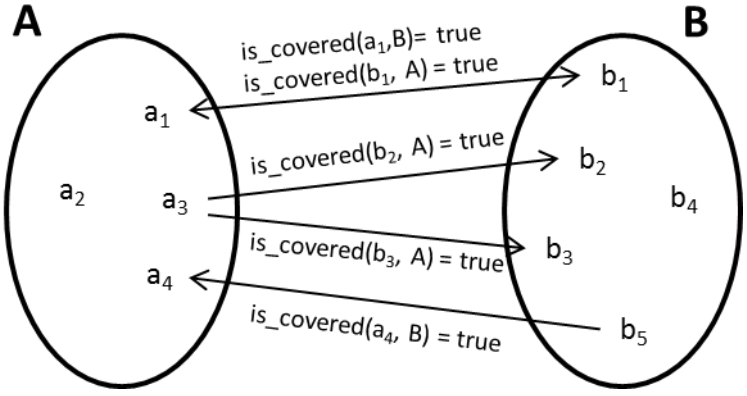

Figure 9: Illustration of coverage

Definition 2. *Set of intentional elements covered*

Given two sets of intentional elements *IE$_a$* and *IE$_b$*, the set of elements from the first set *IE$_a$* covered by the second *IE$_b$* is defined as:

$$coveredIEs(IE_a, IE_b) = \{ie \in IE_a : is\_covered(ie, IE_b)\}$$

In Figure 9, the set of elements of *A* covered by *B* is $\{a_1, a_4\}$ while the set of elements of *B* covered by *A* is $\{b_1, b_2, b_3\}$. Therefore in terms of Definition 2, we have: $coveredIEs(A, B) = \{a_1, a_4\}$ and $coveredIEs(B, A) = \{b_1, b_2, b_3\}$.

The following definition measures to which extent a set of IEs (e.g. *A* in Figure 9) is covered by another (e.g., *B* in Figure 9).

Definition 3. *Set of intentional element coverage*

Given two sets of intentional elements IE_a and IE_b, the coverage of IE_a by IE_b is defined as the percentage of elements in IE_a that are covered by the elements in IE_b:

$$coverage(IE_a, IE_b) = (||coveredIEs(IE_a, IE_b)|| / ||IE_a||) * 100, \text{ being } ||S|| \text{ the size of S}$$

In Figure 9, we have that the percentage of elements of *A* covered by *B* is 50%, whilst the percentage of elements of *B* covered by *A* is 60%.

Given these auxiliary definitions, it is easy to define both concepts of coverage. Given the actor $a_{org} = (name_{org}, IE_{org}, IEL_{org})$, corresponding to the organization in the organizational model, and $a_{str} = (name_{org}, IE_{str}, IEL_{str})$, the actor representing the OSS adopter organization in an OSS adoption strategy model, we define the following:

Definition 4. *Organizational model coverage*

We define the organizational model coverage as the result of:

$$coverage(IE_{org}, IE_{str})$$

Definition 5. *Adoption strategy requirements coverage*

Given ie_{str} the intentional element representing the adoption strategy requirements central task, $ie_{str} \in IE_{str}$, we define the adoption strategy requirements coverage as the result of:

$$coverage(requirements\ (ie_{str}, IEL_{str}), IE_{org}),$$

where $requirements(ie, IEL) = \{x \mid x \in descendants(ie, IEL) \wedge type(x) \in \{task, resource\}\}$

6.2. Merging the Organizational model with the Adoption Strategy model

Once the organization has decided which OSS adoption strategy is going to adopt, the process of refining the organizational model according to the strategy consists on the following steps:

1. Merging both models applying the organizational model coverage, including the elements from the selected OSS adoption strategy model into the organizational model.
2. Making the necessary adjustments to the resulting model in order to adapt the general OSS adoption strategy model to the specific case.

Having the organizational model and the selected strategy model, the final organizational model should be refined adding the strategy model elements. We use the

organizational model coverage in order to include the connection among all the elements from both models. Given $ie_{org} \in IE_{org}$ and $ie_{str} \in IE_{str}$:

- **CC1.** sat(ie_{org}) ⇔ sat(ie_{str}). In this case, ie_{org} is kept and ie_{str} is discarded. The descendants of ie_{str} must be included in the organizational model too to make explicit the ways in which ie_{org} may be satisfied using the adopted strategy.
- **CC2.** sat(ie_{str}) ⇒ sat(ie_{org}). This means that ie_{str} is a means to satisfy ie_{org} in the organization. Therefore, an *i** means-end link may be introduced from ie_{str} (means) onto ie_{org} (end).
- **CC3.** sat(ie_{org}) ⇒ sat(ie_{str}). Even if the organizational element is not covered, we can represent this relationship using the *i** task-decomposition link, in which ie_{str} is a subtask of ie_{org}.
- **CC4.** There are no relationships between the elements. ie_{org} is kept as in CC 1, but no option to add descendants exists.

Figure 10 is used to illustrate the coverage cases and the merging result, using the coverage of the ACME organizational model by the OSS acquisition strategy model that we are using as example in this section (see Table 5). The figure represents two subsets of their intentional elements for actors *ACME* and *OSS Adopter Organization* including the coverages shown in Table 5, such that *Maintain RR Locator* is equivalent to *Act-Maint-Prod* (CC1), the satisfaction of *Acquire OSS component* implies the satisfaction of *OSS component integrated* (CC2), and, finally, the satisfaction of *Develop RR* implies the satisfaction of *Act-INT* and *Act-Test-Prod* (CC3). Descendants of the strategy model elements are not shown in the result for the sake of brevity.

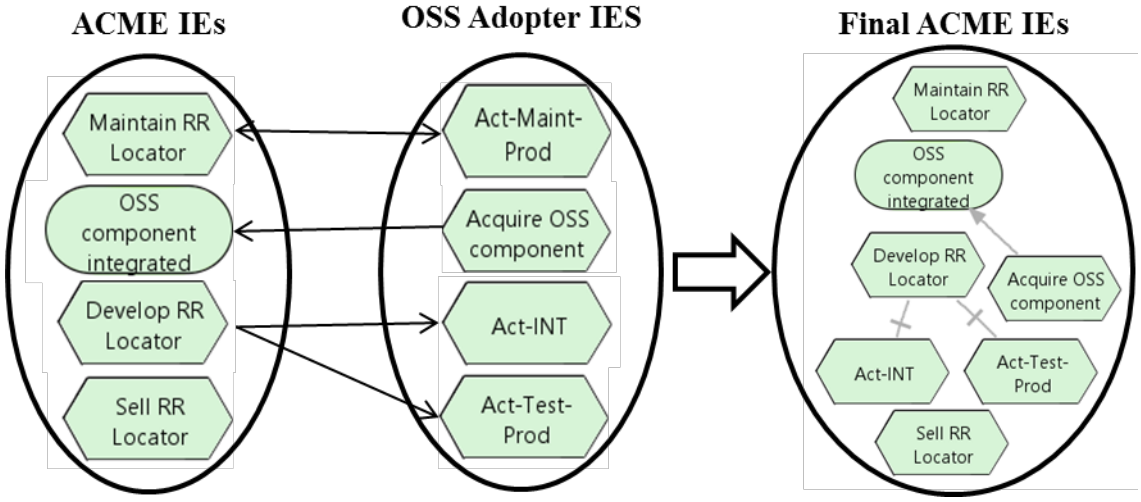

Figure 10. Illustration of model merge (excerpt of the ACME example)

In cases CC2 and CC3, the required connection sometimes cannot be directly provided using a single *i** IE link. The final model needs to include some auxiliary IEs to get a syntactically correct connection. For example, if we have a set of tasks (*t1*, *t2*) in the strategy model covering one goal in the organization model (*g1*) then, as we cannot

connect directly a set of tasks to a goal using task-decomposition links, it is necessary to introduce an intermediate task *t3* as shown in Figure 11 (left).

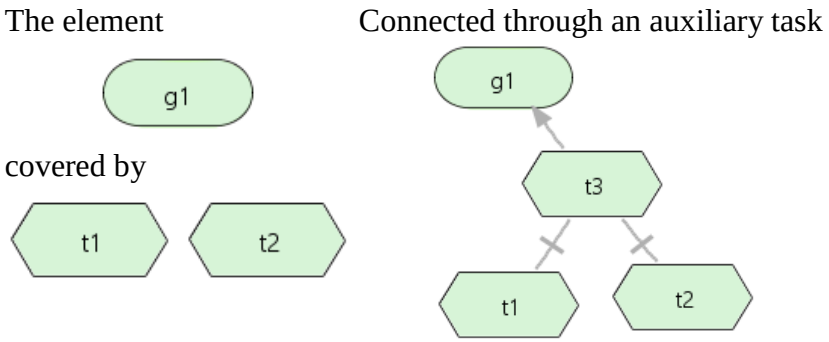

Figure 11: IE coverage with an auxiliary element

To complete the example, we present in Figure 12 the complete model for the ACME company after the merging, as well as the adjustments mentioned at the beginning of the section. In the case that we are using as example in this section, the organization *ACME* is acquiring the component *OSS GIS* to be integrated in its product *RR Locator* as shown in Figure 8. The elements coming from the OSS acquisition strategy model (Figure 2) are shown in italics. The task *Acquire OSS Component* and its decomposition are included as the means to achieve the goal *OSS component integrated*, except for the task *Act-DEP*, it does not appear in the model because the organization uses the OSS component integrating (*Act-INT*) it in its own software (*RR Locator*). The tasks *Act-MAINT-Prod* and *Act-TEST-Prod* from the strategy model are not included because they are covered by *Maintain RR Locator* and *Develop RR Locator* from the organizational model respectively. The *ACME* organization adheres to the strategic goal *Minimize OSS involvement*, but not to *Do not care OSS evolution for maintenance.*

Notice that the task-decomposition links defined for the coverage of the task *Develop RR Locator* by *Act-INT* and *Act-Test-Prod* (see Figure 10) are not included in the final ACME organizational model (Figure 12). After the merge, these links have been removed because they are redundant with having that *Develop RR Locator* has as sub-task the goal *OSS component integrated*, and both tasks belong to this goal decomposition.

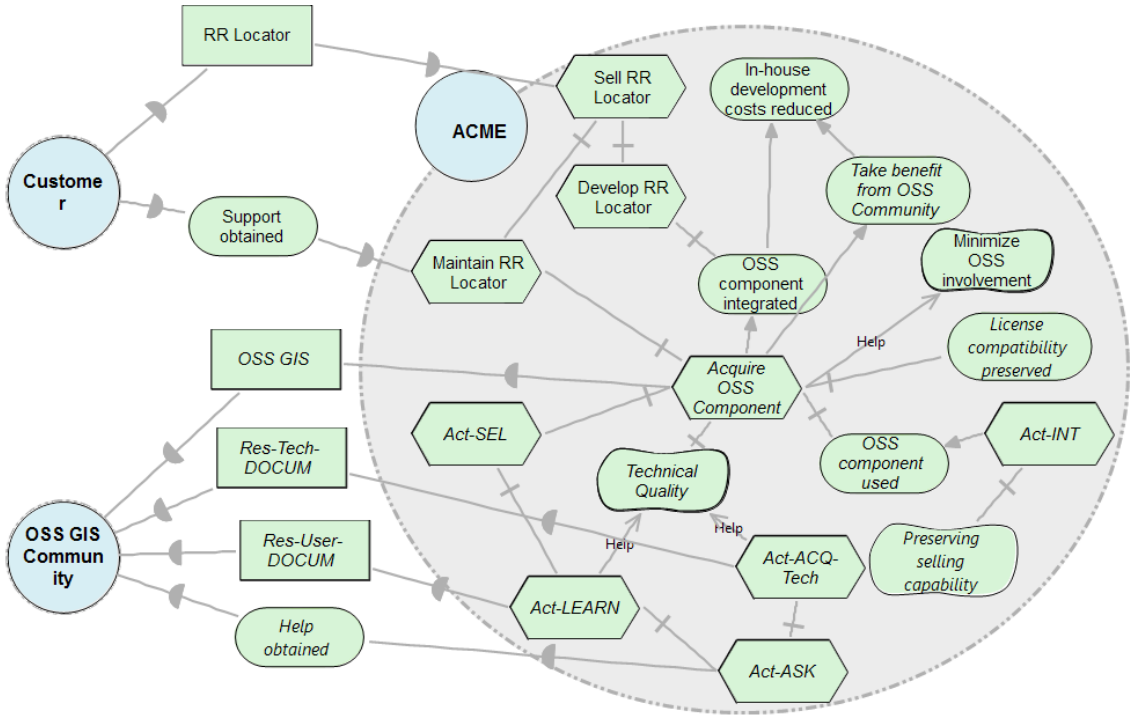

Figure 12: Organizational model adhering OSS acquisition adoption strategy

7. The TEI Case

In this section we show the application of the OSS adoption strategy models to TEI, one of the RISCOSS project industrial partners. TEI is part of Ericsson, one of the world's leading telecommunication corporations. Ericsson produces hardware (telecommunications infrastructure and devices) as well as the software to run it. The company's mission is to empower people, business and society at large, guided by a vision of a sustainable networked society. One of TEI's roles within the Ericsson ecosystem is to provide OSS alternatives to support efficient third party products handling. Therefore, it is important for TEI to adopt OSS components following the adoption strategy that is most suitable to the organization needs.

Figure 13 shows an excerpt of the TEI organizational model related to the maintenance of products including some OSS component, and how its goals impact on Ericsson's goals. Besides the typical strategic goal of any organization for reducing costs (goal *Cost reduced*), the main objectives for TEI are fulfilling the Ericsson's customers' requirements (softgoal *Product requirements achieved*) using *Maintainable code* in order to secure the *Quality of code*. For TEI it is crucial to use *Mature technology* and *Secure code*. When TEI decides to use an OSS component, there are three possibilities for maintaining this code: they can assume the activity (*Provide in-house maintenance*), *Rely on the OSS community for maintenance* or *Contract 3PP* (third party) *organization for maintenance*. For this portion of the TEI's organizational model, the impacted Ericsson goals are *Time-to-market reduced* and *Reputation kept*. Ericsson expects from TEI that

the *Development time is reduced*, *Responsiveness* and *Reliable products* for achieving its goals. Notice that the model only includes the third party organization (*3PP OSS Provider*) in order to illustrate that the maintenance is outsourced; for the sake of brevity, the description of this relation is not exhaustive and not all dependencies between both organizations are included in this model.

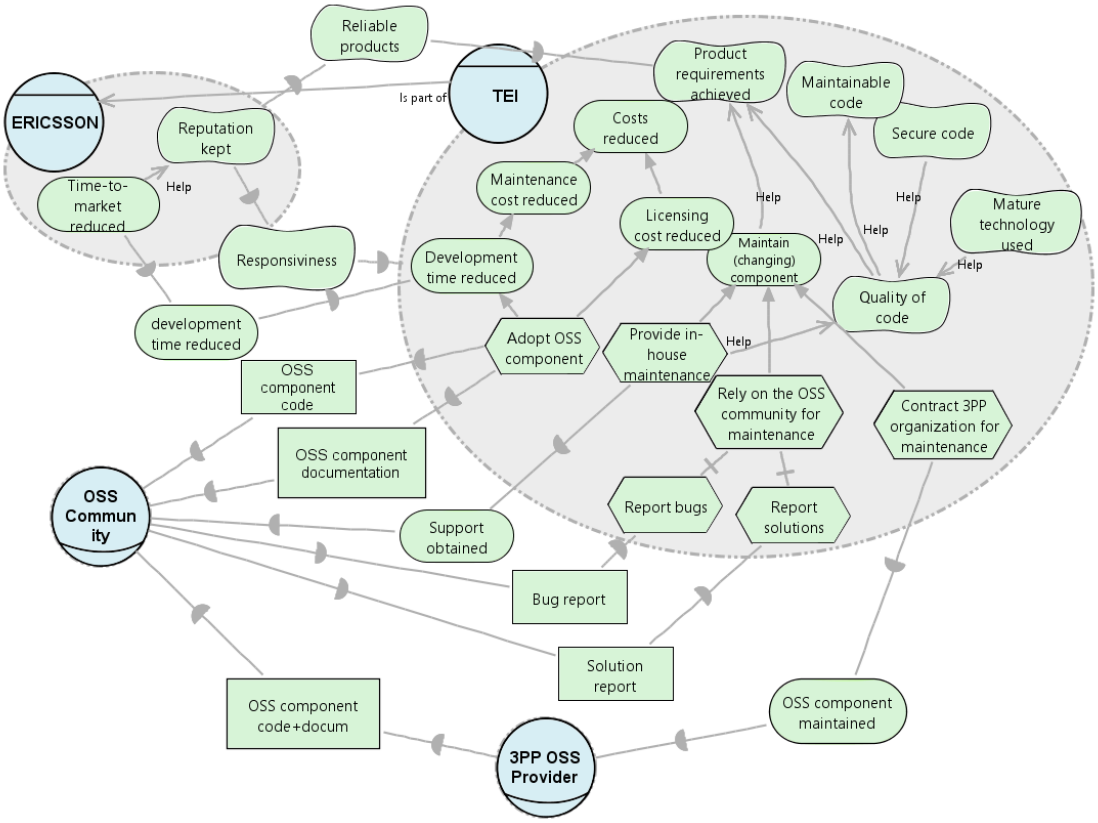

Figure 13: Excerpt of TEI organizational model

The model in Figure 13 synthesizes all the elements from the TEI organizational model that can be considered relevant when TEI interacts with an OSS community for adopting OSS. Therefore, it is an adequate model to be used as a proof of concept of how our proposed OSS adoption strategy models can be applied in an organization.

Table 8 shows the coverage of this abridged TEI organizational model (Figure 13) by the OSS integration strategy model (Figure 3) (organizational model coverage). We can see that the coverage is quite good (68,75% of elements covered). The strategy covers a good number of TEI goals and tasks related to reducing maintenance costs and development time as can be seen in Table 8. Furthermore, we can observe that all goals that remain uncovered except one (*Product requirements achieved*, *Maintainable code*, *Secure code*, *Mature technology used*) depend on the specific OSS component that has been chosen and not on the adoption strategy applied by TEI (an adoption strategy cannot cover them). The remaining goal (*Contract 3PP organization for maintenance*) is, again,

a goal that cannot be covered by any OSS adoption strategy because it consists on contracting an external 3PP company for assuming the maintenance which is an activity not addressed by OSS adoption strategies. We can conclude that the integration strategy fits as well as possible TEI organizational needs and it is a good choice for TEI.

Table 8: TEI organizational model coverage

TEI OSS Organizational model	Coverage	Coverage by OSS integration strategy model (strategy IEs appear underlined)
Product requirements achieved	CC4	Missing
Maintainable code	CC4	Missing
Secure code	CC4	Missing
Mature technology used	CC4	Missing
Costs reduced	CC2	Benefit from co-creation taken ⇒ Costs reduced
Maintenance cost reduced	CC2	Benefit from co-creation taken ⇒ Maintenance cost reduced
Development time reduced	CC2	OSS component used ⇒ Development time reduced
Adopt OSS component	CC2	Integrate OSS component ⇒ Adopt OSS component
Licensing cost reduced	CC2	Integrate OSS component ⇒ Licensing cost reduced
Maintain (changing) component	CC2	Act-MAINT-Prod ⇒ Maintain (changing) component
Provide in-house maintenance	CC2	Act-MAINT-Prod ⇒ Provide in-house maintenance
Rely on the OSS community for maintenance	CC2	Act-MAINT-Prod ⇒ Rely on the OSS community for maintenance
Contract 3PP organization for maintenance	CC4	Missing
Report bugs	CC1	Act-RepBUG ⇔ Report bugs
Report solutions	CC1	Act-RepPATCH ⇔ Report solutions
Quality of code	CC1	Technical Quality ⇔ Quality of code

Table 9 shows the coverage of the OSS adoption strategy requirements. The coverage is incomplete (61,53% of requirements covered). The requirements related to achieve the technical skills in order to use the component (*Act-ACQ-Tech*, *Act-LEARN* and *Act-ASK*) are missing. However, in this case, any organization that wants to use an external component (OSS or not) is willing to acquire the necessary knowledge to use it. Therefore, we realize that these activities, although omitted in our initial TEI organizational model, are actually developed by TEI and thus the TEI organizational model has to be refined by adding them. For the two remaining uncovered requirements (*Act-SUPP* and *Act-ACQ-Man*), we asked TEI representatives whether the company

would be willing to satisfy them. We found that they had actually knowledge about the community practices and applied it when interacting with the community (so their organizational model has to be extended to include *Act-ACQ-Man*). Finally, there was one requirement (*Act-SUPP*) currently uncovered. Since this requirement is one means for the goal *OSS Community Contributed* among two others which are covered by TEI, we conclude that it does not prevent TEI from adopting the integration strategy.

Table 9: OSS integration strategy requirements coverage

OSS Integration strategy	Coverage	Coverage by the TEI OSS Organizational Model (strategy IEs appear underlined)
Requirements		
Act-SEL	CC2	Adopt OSS component ⇒ Act-SEL
Act-ACQ-Tech	CC4	Missing
Act-LEARN	CC4	Missing
Act-ASK	CC4	Missing
Act-DECIDE-Wishlist	CC2	Maintain (changing) component ⇒ Act-DECIDE-Wishlist
Act-DEP	CC2	Adopt OSS component ⇒ Act-DEP
Act-INT	CC2	Adopt OSS component ⇒ Act-INT
Act-MAINT-Prod	CC2	Provide in-house maintenance ⇒ Act-MAINT-Prod Rely on the OSS community for maintenance ⇒ Act-MAINT-Prod
Act-RepBUG	CC1	Report bugs ⇔ Act-RepBUG
Act-PATCH	CC2	Report solutions ⇒ Act-PATCH
Act-RepPATCH	CC1	Report solutions ⇔ Act-RepPATCH
Act-SUPP	CC4	Missing
Act-ACQ-Man	CC4	Missing

Figure 14 depicts a portion of the new organizational model obtained by adhering TEI to the integration adoption strategy. The complete model is not shown for the sake of brevity. This model results from applying the model merging and the needed adjustments as described in Section 6. The new elements that come from the OSS integration strategy model (Figure 3) are shown in italics. Figure 14 includes the portion of the model that appears under the goal *Maintain (changing) component* from the original TEI model (from Figure 13). We can see that TEI keeps its three original alternatives for maintenance: 1) *Provide in-house maintenance*, 2) *Rely on the OSS community for maintenance* and 3) *Contract 3PP organization for maintenance*. Although the first and third alternatives remain the same as in the original model, the second one (consisting on relying on the community behind the OSS component) has been enriched thanks to the integration strategy model. More concretely, the new goal *OSS Community Contributed* now groups all the activities that TEI may carry out as part of its co-creation with the OSS

community. This goal can be achieved thanks to several means: some were already present in the original TEI model (*Report bugs*, *Report solutions*) but the task *Act-SUPP* (that consists on giving support to the community by sponsoring events, etc.) has emerged as a new way to contribute to the community according to the integration adoption strategy. The model also includes new softgoals from the integration strategy, e.g. softgoal *OSS Comp Evolves towards Desired Features* makes now evident the fact that TEI contributions to the OSS community help in making the OSS component evolve towards the features needed by TEI. Finally, the model shows new dependencies that were hidden in the original model, e.g. dependency with dependum *Quality of the Evolved OSS Component* points out that TEI depends on the community for the quality of the evolved component when they rely on the OSS community for maintenance.

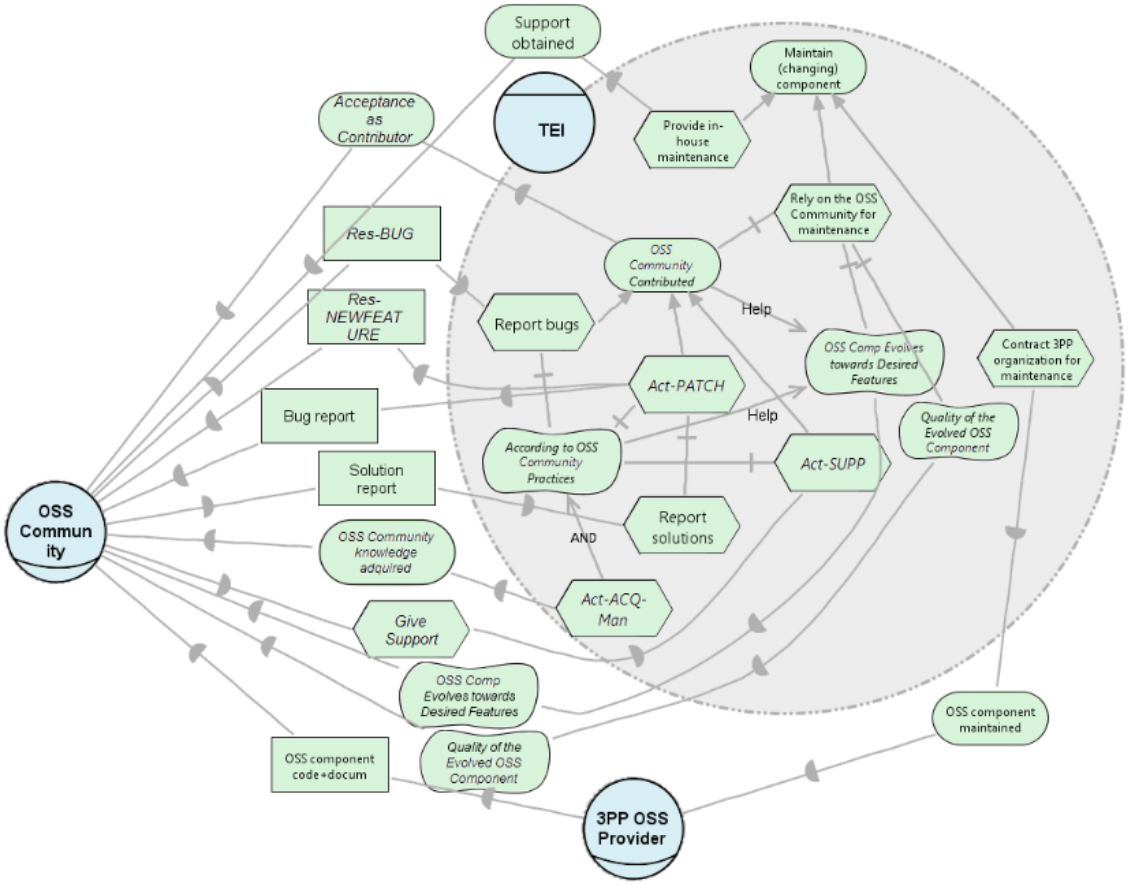

Figure 14: Portion of TEI organizational model adhering integration adoption strategy

8. Conclusions and future work

In this paper, we have embraced a model-based perspective to assess OSS adoption by IT organizations. Goal models in the *i** language have been proposed for adoption strategies models. Three research questions have been investigated:

- ***RQ1****: How to characterize OSS projects?* We have built an ontology to define the activities and resources implied in OSS adoption strategies. As result of the literature review we undertook, we observed that there are several ontologies for OSS but they focus on the perspective of the community, with special attention to: roles of developers (committers, contributors,…), licenses, etc. In the context of business strategies for organizations, this perspective is not the right one, since the needs are different. Therefore, our ontology goes beyond the state of the art and makes possible its use in other works that may be interested in the adopting organization perspective.
- ***RQ2****: How do OSS elements map to OSS adoption strategies?* For each OSS adoption strategy, we have assigned these activities and resources to the actor that is in charge: the adopting organization or the OSS community. As a result, we have provided a characterisation of each strategy in terms of activities undertaken, resources provided, and dependencies of these two actors on each other. Finally, we have designed a set of models for the different strategies expressed in the *i** language.
- ***RQ3****: How do OSS adoption strategies relate to the goals of the OSS adopter organization?* The concept of coverage has been formulated to explore the adequacy of an adoption strategy for a particular organization. The use of goal models and the coverage concept provides an accurate way to assess the right selection of adoption strategy and the update of organizational models to reflect the use of the adoption.

The resulting approach has been satisfactorily applied to the TEI case which has shown the adequacy of the proposal in an industrial setting. From this experience, we have learned that a successful application of the approach requires a good expertise of the modellers applying it. They need a deep understanding both of the organizational model and the OSS adoption strategy models in order to be able to decide about intentional elements coverage by models. Furthermore, the involvement of the organization business managers is required since strategic decisions may have to be taken when merging the organizational model and the OSS adoption strategy model as the TEI case has shown (e.g. deciding to cover an initially uncovered requirement). Although we may expect other cases to be more complex than the TEI case, the portions of the organizational models involved in OSS adoption are not expected to grow proportionally, supporting then scalability of the approach. Of course, further validation of this statement is required.

This work represents a significant extension of a previous paper [24]. More precisely, the extensions are:

- We have extended the ontology by providing properties in addition to the concepts.

- We have presented the models for the six OSS adoption strategies (only two were modelled in [24]).
- We have defined in detail the concept of coverage, which was only sketched in [24].
- We have presented an example of application in a real company.
- We have provided more details in the background section.

Future work goes along several directions. First, we need to work further in the link among business models and OSS adoption strategies, so that the process that has been depicted in Section 6 becomes more prescriptive. Second, concerning the models for the adoption strategies, we want to use *i** roles as a way to organize the ontology elements: roles like Contributor, Governance Body, OSS User, etc., may arrange the different activities and resources, and then the adoption strategies will simply put together the indicated roles in each case. Third, we need to be able to combine OSS adoption strategy models, either because more than one strategy applies at the same time for the same OSS component, or because more than one OSS component is being integrated in a project.

Last, we need to develop the summative evaluation stage of our research approach described in Section 3. In other words, the framework must be applied in the context of other industrial organizations besides the RISCOSS partners.

Acknowledgments

This work is a result of the RISCOSS project, funded by the EC 7th Framework Programme FP7/2007-2013, agreement number 318249. It was also supported by the Spanish project EOSSAC (TIN2013-44641-P).

References

[1] M. Driver, Hype Cycle for Open-Source Software, Technical Report, Gartner, 2013.
[2] H. Chesbrough, Open Business Models: How to thrive in the new Innovation Landscape, Harvard Business School Press, 2006.
[3] D.J. Teece, Business Models, Business Strategy and Innovation. In Long Range Planning Journal, 43 (2010) 172-194.
[4] F. Kudorfer, J.P. Laisne, S. Lauriere, J. Lichtenthaler, G. Lopez, C. Pezuela, State of the art concerning business models for systems comprising open source software. QualiPSo deliverable, 2007.
[5] C. Daffara, Business models in FLOSS-based companies. In Conference on Open Source Systems (OSS) Workshops, 2007.
[6] S. Lakka, T. Stamati, C. Michalakelis, D. Martakos, The Ontology of the OSS Business Model: An Exploratory Study, International Journal of Open Source Software and Processes, 3 (2011) 39-59.
[7] A. Osterwalder, The business model ontology - a proposition in a design science approach, University of Lausanne, PhD Dissertation, 2004.
[8] A. Osterwalder, Y. Pigneur, C. Tucci, Clarifying Business Models: Origins, present and Future of the Concept. In Communitacions of the Association for information Systems, 15 (May 2005).
[9] V. Chang, H. Mills, S. Newhouse, From Open Source to long-term sustainability: Review of Business Models and Case studies. In All Hands Meeting 2007, OMII-UK Workshop.

[10] C. Daffara, Business models in FLOSS-based companies.. In Open-Source Software in Economic and Managerial Perspective (OSSEMP) 2007.
[11] A. Dornan, The Five Open Source Business Models. In Information Week 2008.
[12] E. Yu, Modelling Strategic Relationships for Process Reengineering, PhD. thesis, Toronto, 1995.
[13] X. Franch, A. Susi, M. C. Annosi, C. P. Ayala, R. Glott, D. Gross, R. S. Kenett, F. Mancinelli, P. Ramsamy, C. Thomas, D. Ameller, S. Bannier, N. Bergida, Y. Blumenfeld, O. Bouzereau, D. Costal, M. Dominguez, K. Haaland, L. López, M. Morandini, A. Siena, Managing Risk in Open Source Software Adoption. In International Joint Conference on Software Technologies (ICSOFT), 2013, pp. 258-264.
[14] B. Kitchenham, Guidelines for performing Systematic Literature Reviews in Software Engineering v2.3., EBSE Technical Report EBSE-2007-01, 2007.
[15] C. Ayala, L. López, D1.3 Modeling Support (consolidated version), Technical Report, RISCOSS FP7 project, 2014.
[16] A. Ankolekar, K. Sycara, J. Herbsleb, R. Kraut, C. Welty, Supporting online problem-solving communities with the semantic web. . In International Conference on World Wide Web (WWW), 2006, pp. 575-584.
[17] G.L. Simmons, T.S. Dillon, Towards an Ontology for Open Source Software Development. In International Conference on Open Source Systems (OSS), 2006, pp. 65-75.
[18] T.S. Dillon, G.L. Simmons, Semantic web support for open-source software development. In Signal Image Tecnology and Internet Based Systems International Conference (SITIS), 2008, pp. 606-613.
[19] I. Mirbel, OFLOSSC, an ontology for supporting open source development communities. In International Conference on Enterprise Information Systems (ICEIS), 2009, pp. 47-52.
[20] X. Franch, On the Lightweight Use of Goal-Oriented Models for Software Package Selection. In International Conference on Advanced Information Systems Engineering (CAiSE), 2005, pp. 551-566.
[21] C. Rolland, Requirements Engineering for COTS based Systems, In International Journal of Information and Software Technology, 1999, 41 (14) 985-990.
[22] P. Soffer, B. Golany, D. Dori, Aligning an ERP system with enterprise requirements: An object-process based approach. In Computers in Industry, 56 (2005) 639–662.
[23] L. López, X. Franch, J. Marco, Specialization in *i** Strategic Rationale Diagrams. In International Conference on Conceptual Modelling (ER) 2012, pp. 267-281.
[24] L. López, D. Costal, C. P. Ayala, X. Franch, R. Glott, K. Haaland, Modelling and Applying OSS Adoption Strategies.. In International Conference on Conceptual Modelling (ER) 2014, pp. 349-362.