Generating Win-Win Strategies for Software Businesses Under Coopetition: A Strategic Modeling Approach

Vik Pant[1(✉)] and Eric Yu[1,2]

[1] Faculty of Information, University of Toronto, Toronto, Canada
vik.pant@mail.utoronto.ca, eric.yu@utoronto.ca
[2] Department of Computer Science, University of Toronto, Toronto, Canada

Abstract. Interorganizational coopetition describes a phenomenon in which businesses cooperate and compete simultaneously. Such behavior is commonplace among software firms wherein vendors concomitantly deal with each other both as partners and as rivals. Sustainable coopetitive relationships are predicated on the logic of win-win strategies. Conversely, win-lose or lose-lose strategies do not lead to durable coopetitive relationships. This aspect of coopetition requires decision-makers in coopeting software businesses to generate and analyze win-win strategies. This paper proposes a strategic modeling approach to systematically search for alternatives and generate win-win strategies. This approach synergistically combines *i** goal-modeling to analyze the distributed intentional structures of actors and Game Tree decision-modeling to reason about the moves and countermoves of actors. An illustrative example of a published case study is presented to demonstrate the strengths and weaknesses of this methodology.

Keywords: Strategic modeling · Coopetition · Win-win · Positive-sum

1 Introduction

Many software businesses join ecosystems (SECOs) to benefit from open innovation [1] as well as to access: shared market, common technological platform, and opportunities for information/resource/artifact exchange [2]. Each SECO comprises an intricate network of multifaceted relationships among software vendors. Coopetition, which refers to simultaneous cooperation and competition among two or more actors [3], is commonplace within SECOs [4].

The need for analyzing strategic relationships in and among SECOs has been by emphasized by several researchers [2, 5, 6]. Many researchers have proposed SECO modeling techniques to explain structures and processes of SECOs [7–11]. However, none of these SECO modeling techniques focus directly on coopetition in and among SECOs.

Decision-makers require insight into the intentions of coopeting actors to discern the motives behind their actions and responses. They also require foresight to predict the moves and countermoves of actors under coopetition. Game Trees (i.e., multi-actor Decision Trees) are commonly used to analyze multi-actor decisioning scenarios.

K. Wnuk and S. Brinkkemper (Eds.): ICSOB 2018, LNBIP 336, pp. 90–107, 2018.
https://doi.org/10.1007/978-3-030-04840-2_7

However, Pant and Yu note, "Game trees elide the intentional structure of the players" [12]. This is because while Game Trees encode the motivations of actors into *payoffs* implicitly they do not express those motivations explicitly. Therefore, Game Trees do not provide a systematic method for exploring the space of potential strategic alternatives to generate new win-win strategies.

*i** Strategic Rationale (SR) models can be used to show the internal intentional structures of actors overtly and can be used to complement Game Trees. A novel methodology for the synergistic use of actor goal modeling (with *i**) and decision modeling (with Game Trees) was introduced by Pant and Yu [12]. The present paper extends that work by proposing a systematic method for generating win-win strategies.

The remainder of this paper is organized as follows. In the next section we discuss strategic outcomes in coopetitive relationships including the notions of win-win, win-lose, and lose-lose strategies. The third section presents a modeling-based methodology for generating and evaluating win-win strategies among actors by building upon a novel approach introduced in [12]. In the fourth section we instantiate this methodology by applying it to a published case study about SECOs under coopetition. In the fifth section we review related work while in the sixth section we discuss our conclusions and future work.

2 Strategic Outcomes in Coopetitive Relationships

Simultaneous cooperation and competition is characterized by the partially congruent interest structures of coopeting actors [13]. Actors in such relationships "cooperate to grow the pie and compete to split it up" [14]. According to Game Theory (e.g., [3]), a multi-actor relationship can be classified as: positive-sum, zero-sum, or negative-sum.

In positive-sum scenarios, each actor gains by participating in the relationship; in zero-sum scenarios, some actors are better off while some actors are worse off by participating in the relationship; and in negative-sum scenarios all actors are harmed by participating in the relationship. In zero-sum scenarios, the magnitude of gain for some of the actors equals the degree of pain for the other actors in that relationship.

It is definitional and logical that rational and self-interested actors are likely to voluntarily take part only in those relationships that are beneficial for themselves (i.e., zero-sum but only where they are advantaged, or positive-sum) [15].

Coopetitive relationship are regarded as strategic because the actions of any actor can impact the actions of any other actor(s) and, similarly, the decisions of any actor can inhibit or impel the decisions of any other actor(s). Therefore, actors in such relationships are codependent on each other for the achievement of their common goals as well as individual objectives.

A win-win strategy is the sole practical choice for an actor under coopetition because only it is likely to yield an equilibrium condition under which all actors are willing to remain in that relationship voluntarily [3]. Therefore, decision-makers in coopetitive organizations must search for win-win strategies by: (1) analyzing existing alternatives, and (2) generating new alternatives. This can be done by using *i** to search for new alternatives and Game Trees to evaluate those alternatives. Complementary usage of *i** and Game Trees is demonstrated in Sects. 3 and 4 where these techniques are used to search for new alternatives. The process for modeling, evaluating, and exploring the space of alternatives is depicted in Fig. 1.

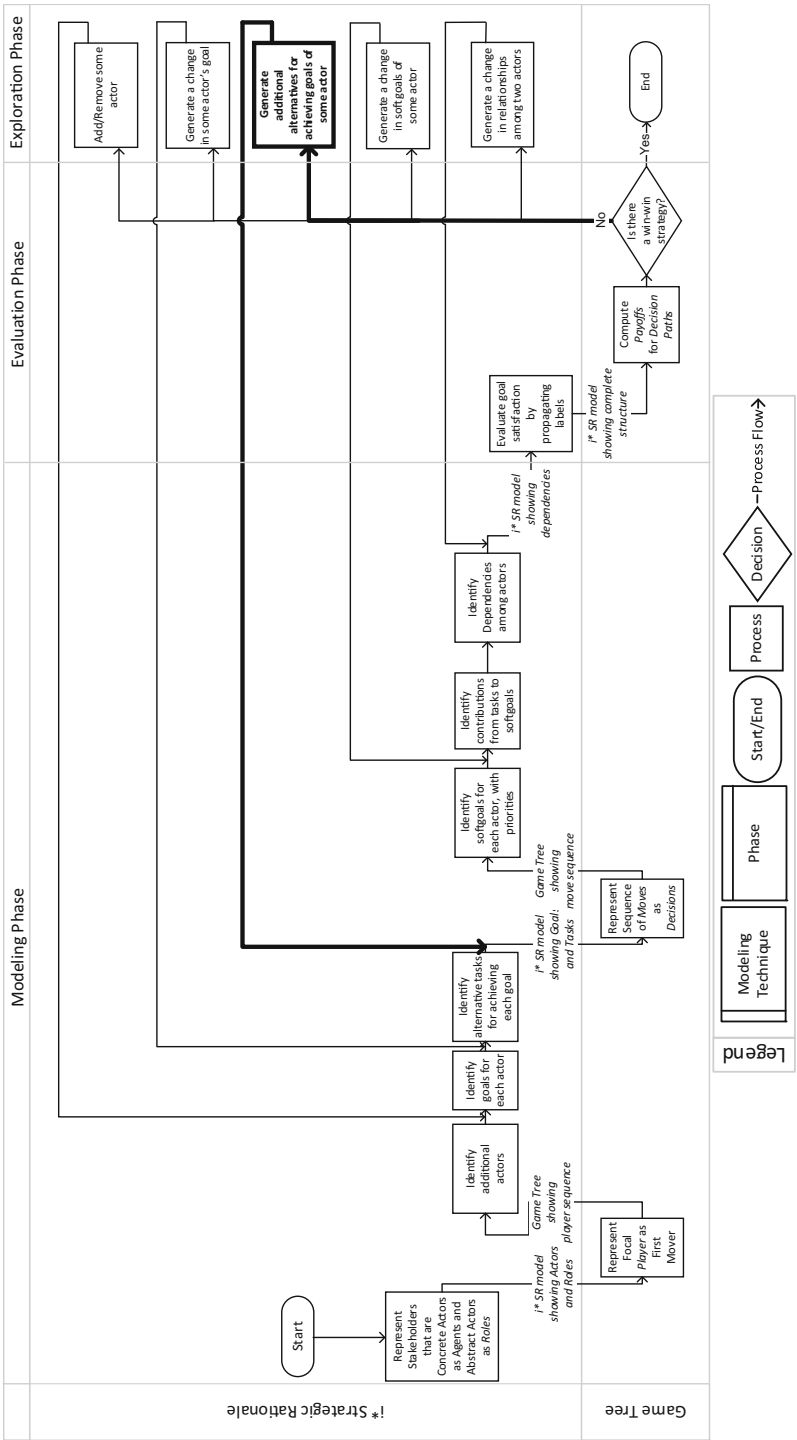

Fig. 1. Process to develop *i** SR diagram and its corresponding Game Tree [**emphasis** on introduction of new task] (adapted from [12])

3 Methodology for Generating Win-Win Strategies with *i** and Game Trees

We explain a methodology for generating win-win Strategies with *i** and Game Trees by using a simple example from Game Theory. Let us assume that two siblings, namely Cake Cutter (CC) and Slice Selector (SS), wish to divide a cake among themselves. The only rule that governs their sharing of a cake is that one sibling cuts the cake (CC) into two slices and the other sibling distributes each of those slices (SS). Researchers from myriad disciplines have contemplated concepts such as fairness and reciprocity using variations of this basic scenario [16–20].

Suppose that both CC and SS wish to obtain the larger share of cake for themselves and that CC has only one alternative available to it which is of cutting the cake into two unequal slices. Consequently, SS has two alternatives available to it which are that it can either take the larger slice or the smaller slice for itself and give the remaining slice to CC.

If SS takes the larger slice then its goal is satisfied but the goal of CC is denied. Alternatively, if SS takes the smaller slice then its goal is denied but the goal of CC is satisfied. Therefore, cutting the cake into unequal slices by CC does not lead to a positive-sum outcome. Moreover, if a decision by CC to cut the cake into unequal slices can lead to SS winning and CC losing then these alternatives represent a win-lose strategy.

CC must generate one or more new alternatives for achieving its goal since the existing alternative does not represent a win-win strategy. CC can generate a win-win strategy by analyzing its own alternatives and goals as well as those of SS. A new alternative that CC can generate is to cut the cake into equal slices. This new alternative for CC necessitates SS to generate a new alternative as well. This is because there is no such thing as a larger or a smaller slice when the cake is cut into equal slices. Therefore, the new alternative for SS is to take either of the equal slices. This allows both CC and SS to obtain equal slices. Considering the rules of their arrangement this allows both to satisfy their goals.

Formal solutions to such fair-division problems (e.g., “I cut, you choose”) have been proven via minimax and maximin theorems [21]. Game Trees are commonly used to analyze such scenarios because they support the notion of *payoffs*. However, Game Trees do not allow a systematic search for new alternatives—which is a necessary step for generating win-win strategies.

Figure 1 presents a structured and systematic methodology for generating win-win strategies among actors. This proposal complements the Game Tree method with a strategic goal-modeling approach. It is explained in this section with reference to this example of cake sharing. Figure 2a is an *i** Strategic Rationale (SR) diagram that depicts the application of the Modeling and Evaluation phases of Fig. 1. *i** (denoting “distributed intentionality”) is a goal- and actor-modeling language that supports strategic reasoning [22].

In the Modeling phase, Fig. 2a portrays the relationship between two actors—CC and SS. Figure 2a shows that the primary objective of the two parties is to get the larger share of the cake for itself. Each actor uses this as a quality criterion to evaluate and

compare alternatives. They assess each option by estimating the impact of an option on their obtaining the larger share of the cake. This quality criterion is depicted as a *softgoal*. The relative importance of each *softgoal* is depicted with one or more exclamation mark(s) to indicate lower (!) and higher (!!) priorities.

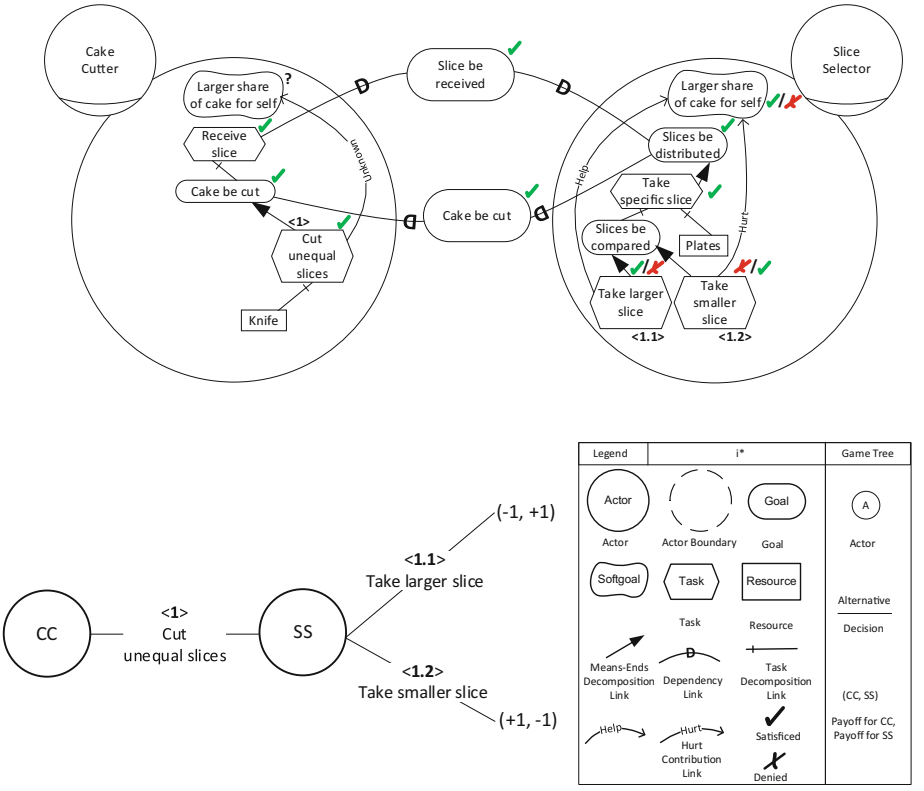

Fig. 2. (a) *i** SR model depicting As-Is relationship among CC and SS. (b) Game Tree depicting As-Is decision alternatives with resulting payoffs

A *task* is an activity that can be used to achieve a *goal*. In the As-Is scenario, CC has one way of achieving the *goal* "Cake be cut", by cutting into unequal slices. We extend the *i** notation slightly to depict multiple options as well as moves and countermoves in the same *i** model. Each option is designated a number which is enclosed within angle brackets. For example, "Cut unequal slices" is identified as <1>. Countermoves corresponding to this option are denoted as <1.x> where x denotes a possible response to "Cut unequal slices". Therefore, "Take larger slice" is denoted as <1.1> and "Take smaller slice" is denoted as <1.2>. Countermoves in response to <1.1> and <1.2> can be depicted as <1.1.x> or <1.2.x> (not shown). This allows a sequence of moves and countermoves of any length to be represented in the *i** model.

Tasks can be refined into lower-level *goals*, *tasks*, *softgoals*, and *resources*. These subsidiary *goals*, *tasks*, *softgoals*, and *resources* are related to a higher-level *task* using a *task decomposition link* such that each of the lower level elements must be satisfied in order for their associated higher-level *task* to be fulfilled. A *resource* (e.g., knife, plate) is a physical or informational entity required to perform a *task.*

A task is related to a goal using a *means-ends link* (with solid arrowhead) such that the completion of any *task* leads to the fulfilment of its associated *goal*. A *goal* represents a state of affairs that an actor wishes to achieve in the world.

Contribution links (e.g., *help*, *hurt, unknown*) (curved arrows with open arrowheads) are used to show the impact of *tasks* and *softgoals* on one or more *softgoals*. *Labels (*such as *satisfied, denied)* are propagated along contribution links to derive the impact of model elements on other elements. In this example, it is *unknown* whether cutting the cake into unequal slices will help or hurt CC's *softgoal* of obtaining the larger share of the cake. This is because, per the rules of their arrangement, it is SS that decides the distribution of cake slices. Therefore, if SS keeps the larger piece for itself (e.g., exhibiting opportunism) then CC's *softgoal* will not be satisfied but if SS keeps the smaller piece for itself (e.g., demonstrating altruism) then CC's *softgoal* will be satisfied.

Figure 2a shows that SS can choose either the larger or the smaller slice for itself and give the other slice to CC. This choice is shown as two alternative *tasks* leading towards the same *goal* via *means-ends* links. SS compares unequally sized slices to decide whether to keep or give the larger or smaller sized slice. This is shown as a *subgoal*. SS judges an alternative by reckoning its ability to help SS obtain the larger share of cake for itself. This is depicted as a *softgoal*.

CC and SS are inter-reliant on each other for the sharing of cake to take place among themselves. In the As-Is scenario, SS needs CC to cut the cake and CC needs SS to obtain a slice of the cake. This inter-dependency among CC and SS is shown via *dependency links*. A *depender* is an actor that depends on a *dependee* (i.e., another actor) for a *dependum* (i.e., something such as a *task* to be completed, a *goal* to be satisfied, a *resource* to be provided, or a *softgoal* to be fulfilled). The curved side on the D in the *dependency link* faces the *dependee* while the flat side faces the *depender*.

We complement *i** means-ends modeling with Game Tree modeling to show the gain or loss associated with various strategies for each *actor/player*. Figure 2b depicts a Game Tree representing sequential *actions/decisions* by CC and SS as well as the *payoffs* associated with each *action/decision path*. Dixit and Nalebuff present an overview of Game Trees in [15].

In the Modeling phase, Fig. 2b shows the sequence of *actions/decisions* by the actors. In the Evaluation phase, the *payoffs* for each configuration of move and countermove for every actor are calculated by assessing *softgoal* satisfaction/denial in Fig. 2a. Figure 2b shows that CC moves first since it is necessary for it to cut the cake before SS can distribute the cake slices. CC has only one strategy available to it in the As-Is configuration. Therefore, CC decides to adopt the strategy of cutting the cake into unequal slices. SS makes the next move by deciding whether to give the larger or smaller of the cake slices to CC. SS can act opportunistically (larger slice for SS) or altruistically (larger slice for CC).

Let us suppose that if SS decides to keep the larger slice of cake for itself then it earns a payoff of +1 while CC earns a payoff of −1. This is because, in this situation, SS is able to satisfy its *softgoal* while CC is unable to fulfil its *softgoal*. Conversely, If SS decides to keep the smaller slice of cake for itself then it earns a payoff of −1 while CC earns a payoff of +1. This is because, in this situation, SS is unable to satisfy its *softgoal* while CC is able to fulfil its *softgoal*.

This integrated analysis, of the *i** SR model and Game Tree, indicates that the As-Is relationship between CC and SS only comprises win-lose strategies and not any win-win strategies. This is because in one outcome CC ("+1") is advantaged but SS ("−1") is disadvantaged while in the other outcome SS ("+1") is advantaged but CC ("−1") is disadvantaged. This aspect of the As-Is relationship between CC and SS motivates the need for generating win-win strategies by applying the steps recommended in the Exploration phase.

In the Exploration phase, one or more subject matter experts (SME) or domain specialists contemplate ideas for generating win-win strategies. They follow an iterative and incremental process for enlarging and pruning the *i** models and their associated Game Trees.

The Exploration phase consists of five steps that are arranged in a non-deterministic manner. SMEs can choose to start with any of the steps in the Exploration phase. Each step in the Exploration phase loops back to a corresponding step in the Modeling phase. This allows SMEs to make one change at a time to the intentional (*i**) model and assess its impact on the decision-support (Game Tree) model in an incremental fashion. SMEs iterate through the Exploration, Modeling, and Evaluation phases until they successfully generate one or more win-win strategies.

In the Exploration phase, SMEs apply their knowledge of the motivations of the actors as well as of their shared context to select any of the steps in that phase. They can change the relationship among two actors by changing the object of their dependency on each other (i.e., *dependum*). Alternatively, they can change a quality criterion (i.e., *softgoal*) by which some actor compares alternate means for achieving their desired ends. Else, they can develop a new alternative (i.e., *task*) for achieving the objectives (i.e., *goal*) of some actor. Or, they can change an actor's objective (i.e., *goal*). Otherwise, they can Add/Remove an actor from the relationship. After making one change at a time the SMEs can repeat the process (Modeling and Evaluation phases) to check whether any win-win strategy is generated from that change.

In our example, we suppose that CC (i.e., a SME of cake sharing) performs the steps in the Evaluation phase to extend and refine Fig. 2a and b. CC does this because the As-Is scenario does not consist of any win-win strategies. To generate a new win-win strategy (i.e., To-Be scenario), CC can begin by selecting any of the steps in the Exploration phase.

Figure 3a (To-Be) is an *i** Strategic Rationale (SR) diagram that extends Fig. 2a (As-Is) by applying steps from the Exploration phase. Figure 3b depicts a Game Tree that extends Fig. 2b to show new *payoffs* associated with an additional strategy that is depicted in Fig. 3a. Model elements with black color represent existing model elements from Fig. 2a and b while model elements with blue color represent new model elements in Fig. 3a and b.

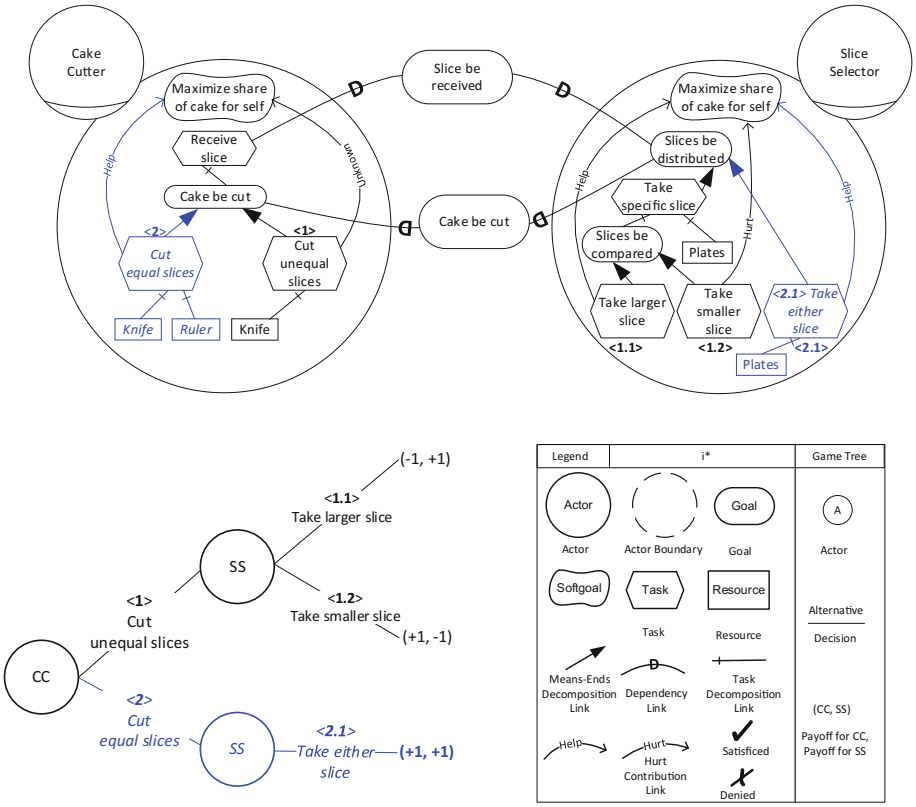

Fig. 3. (a) *i** SR model depicting To-Be relationship among CC and SS. (b) *i** Game Tree depicting To-Be decision alternatives with resulting payoffs

CC evaluates Fig. 2a and b to understand the reasons for the absence of any win-win strategy in the As-Is scenario. CC recognizes that its As-Is strategy of cutting the cake into unequal slices can be disadvantageous for itself. This is because SS has a *softgoal* of maximizing its (SS's) own share of the cake which can only be satisfied if SS selects the larger slice for itself and gives the smaller slice to CC. CC realizes that it is improbable for SS to act altruistically by selecting the smaller slice for itself and giving the larger slice to CC since the *i** model does not contain any *softgoal* to justify such behavior from SS.

CC starts the Exploration phase by contemplating a new alternative that can help it to achieve its sole *softgoal*. However, this alternative must also help SS to satisfy its only *softgoal*. This new strategy (To-Be) can only exist if CC cuts the cake into equal slices. This new alternative for CC will also change the space of alternatives available to SS. This is because by cutting the cake into equal slices CC will require SS to generate a new alternative of taking either slice. This new strategy (To-Be) will bring the interest structures of CC and SS into congruence because it will allow both of them to satisfy their respective *softgoals*.

Figure 3b represents the updated *payoffs* for CC and SS considering this new strategy (To-Be). If CC cuts the cake into equal slices then both CC and SS earn a payoff of +1. This is because the To-Be strategy allows SS to maximize its own share of the cake while also permitting CC to obtain the largest possible share of the cake considering the terms of their arrangement. By generating this new strategy, CC eliminates the possibility for SS to act either opportunistically or altruistically. This new alternative represents a win-win strategy for both CC and SS.

The next section demonstrates the application of this methodology to a real-life historic case. It clarifies the systematic structure of the reasoning steps via instantiations of goal-models (*i** SR diagrams) and complementary decision-models (Game Trees). The following example draws upon multiple published sources [see 23–27]. It is presented as an interpretive reconstruction that interleaves ground truth (i.e., historical fact) and creative conjecture (e.g., new alternatives). It is presented in this way to accommodate and reflect factual and counterfactual aspects of this case.

4 A Case Example of Coopetition: Apple and Adobe SECOs

A widely studied case of industrial coopetition among SECOs pertains to the relationship between Apple and Adobe [see 23–27]. Apple and Adobe operated as partners because Adobe's Flash-based web-applications added value to Apple's web browser (Safari) on its desktop operating system (macOS). Similarly, Adobe generated acceptance and adoption of its Flash technology from Apple's customer base that accessed Flash-based web-applications on their Apple computers. However, Apple and Adobe also behaved as rivals since they operated competing SECOs for mobile apps (i.e., Apple iOS app store and Adobe Flash Gallery).

Figure 4a depicts an *i** SR model of Apple's "walled garden" strategy and Adobe participation. In the Modeling phase, we use *i** to show the internal intentional structures of Adobe and Apple. This model is based on details from [23–27] and is adapted from [12]. The left side of Fig. 4a shows a condensed model of Apple's strategy. Apple's objective was to drive the adoption of its proprietary OS (i.e., iOS) in the mobile device market ("iOS be adopted in smart mobile device market"). The success of iOS was tied to higher sales of iPhone, iPod, and iPad devices because Apple's iOS and its mobile devices were only compatible with each other (not shown).

We extend the notation of *i** slightly to depict the impact of multiple options on *softgoals* in the same *i** model. A *softgoal* satisfaction/denial label is preceded by a number which represents the option that leads to the satisfaction/denial of that *softgoal*. For example, "Reference Objective-C API" is shown as option <2>, which satisfies the *softgoal* titled "Apps be optimized for iOS".

Apple's SECO was a core component of its iOS proliferation strategy. A mobile OS requires a complementary catalog of third-party apps to boost its acceptance and adoption by users ("External innovation be encouraged"). Third-party apps bring new capabilities to a mobile OS and make that mobile OS more useful for its users. Hence, a relatively large catalog of apps ostensibly affords greater choice to the users of a SECO compared to a relatively small catalog.

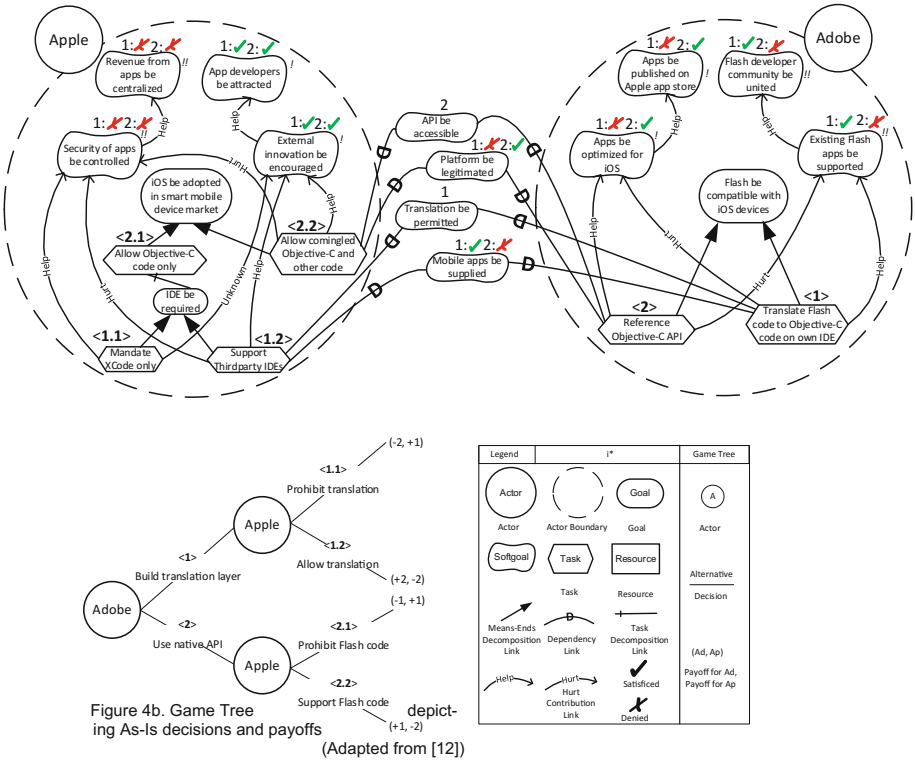

Figure 4b. Game Tree depicting As-Is decisions and payoffs (Adapted from [12])

Fig. 4. (a) *i** SR model depicting As-Is actor relationships. (b) Game Tree depicting As-Is decisions and payoffs (Adapted from [12])

Moreover, positive cross-side network effects synergistically correlate the user base and developer community on a SECO [34] such that growth in the numbers of apps (and their developers) on a SECO attracts more users to that SECO while growth in the number of users on a SECO incents more developers to develop apps for that SECO ("App developers be attracted").

Apple coupled its mobile hardware and software tightly so that it could exert maximal control on the security of apps that were used on iPhone, iPod, and iPad devices ("Security of apps be controlled"). App developers could generate revenues by charging users for downloading their apps in addition to building in-app purchases and value-added offers into their mobile apps (not shown). Apple protected its commissions from these income streams by forcing users to purchase apps from its iOS app store (i.e., prevent revenue flight) as well as requiring developers to use its IDE and programming language (i.e., prevent revenue obfuscation). This "walled garden" strategy helped Apple to safeguard its commissions ("Revenue from apps be centralized").

Apple had two strategic options ("Allow Objective-C code only" and "Allow comingled Objective-C and other code"). Objective-C is Apple's proprietary programming language that is supported by iOS. Each of these options impacted Apple's

softgoals differently. The option to "Allow comingled Objective-C and other code" (e.g., Adobe Flash code) afforded app developers the opportunity to hide forbidden or malicious functionality outside the purview of Apple security reviews (*Hurts* softgoal "Security of apps be controlled").

The option to "Allow Objective-C code only" had two sub-options. Objective-C code could be developed using Apple XCode ("Mandate XCode only") or generated using a third-party IDE ("Support Third-party IDEs"). XCode is Apple's native integrated development environment (IDE) for iOS. Third-party IDEs afforded app developers the opportunity to bypass security policies implemented by Apple in its XCode IDE (*Hurts* softgoal "security of apps be controlled").

The "Mandate XCode only" option could have positive or negative impact ("Unknown") on the *softgoal* "External innovation be encouraged". The outcome of this option depended upon the perceived difficulty of using Apple's XCode IDE by an app developer that was unfamiliar with Objective-C. If usage of XCode was perceived as being simple then it would Help that *softgoal* but if it was perceived as being complex then it would Hurt that *softgoal* (not shown).

Now consider Adobe's strategic options. Adobe intended for its Flash technology to be supported on Apple iOS devices ("Flash be compatible with iOS devices"). A plethora of Flash-based web-apps could be accessed on the Internet and Adobe's goal was to make these apps available on popular mobile devices such as iPhones, iPods, and iPads. To achieve this objective Adobe had two alternatives which were: "Reference Objective-C API" and "Translate Flash code to Objective-C code on own IDE". Each of these strategies had different pros and cons for Adobe.

The first alternative involved translating Flash code into Objective-C code directly within Adobe's IDE for developing Flash applications (Adobe Flash Builder). Under this option, developers of Adobe Flash apps did not need to use any Apple tools or technologies. This translation option is depicted as scenario <1> in Fig. 4a. This option allowed reuse of Flash code (*Helps* softgoal "Existing Flash apps be supported"). It also allowed cohesion to be maintained in the Flash developer community (*Helps* softgoal "Flash developer community be united").

The second alternative involved referencing Objective-C API from Flash code directly within Adobe's IDE for developing Flash applications (Adobe Flash Builder). This commingling option is depicted as scenario <2> in Fig. 4a. This option allowed developers to optimize apps for iOS (*Helps* softgoal "Apps be optimized for iOS") and for those apps to be publishable on Apple iOS app store (*Helps* softgoal "Apps be published on Apple app store").

Adobe depended on Apple for the operationalization of both options under its consideration (i.e., "Translate Flash code to Objective-C code on own IDE" and "Reference Objective-C API"). This reliance is shown via outbound *dependency links* from Adobe to Apple ("Translation be permitted" and "API be accessible" respectively for the two options).

Figure 4b depicts the *payoffs* for Adobe and Apple for each of these scenarios. In the Evaluation phase, we use a Game Tree to compare various alternatives. Adobe was the first-mover since it had the choice of selecting either the translation (<1>) or the commingling (<2>) option. Apple was the second mover since it controlled the iOS platform and could permit or prohibit actions by third-parties that depended on it for

some decision or action. Therefore, Apple could respond to Adobe either by supporting its first-move or blocking it.

If Adobe selected the translation option (<1>) and Apple supported it then Adobe obtained a payoff of +2 while Apple obtained a payoff of −2. This is because the high priority *softgoals* of Adobe were achieved but the high priority *softgoals* of Apple were denied (comparing softgoals priorities and achievements associated with <1> in Fig. 4a). However, if Adobe selected the translation option (<1>) and Apple blocked it then Adobe obtained a payoff of −2 while Apple obtained a payoff of +1. This is because Apple was able to avoid the countermanding of its high priority *softgoals* but the high priority *softgoals* of Adobe were not fulfilled (<1> in Fig. 4a). For the purpose of illustration, we use simple representative values for the payoffs.

Alternatively, if Adobe selected the commingling option (<2>) and Apple supported it then Adobe obtained a payoff of +1 while Apple obtained a payoff of −2. This is because some *softgoals* of Adobe, albeit of lower priority, were satisfied but the high priority *softgoals* of Apple were denied (<2> in Fig. 4a). However, if Adobe selected the commingling option (<2>) and Apple blocked it then Adobe obtained a payoff of −1 while Apple obtained a payoff of +2. This is because high priority *softgoals* of Adobe were unfulfilled but Apple was able to avoid the denial of its high priority *softgoals* (<2> in Fig. 4a).

This analysis of Fig. 4b, following the Evaluation phase of Fig. 1, shows that the relationship between Adobe and Apple did not comprise of any win-win strategies. Rather their relationship characterized only win-lose strategies wherein if one party wins then the other party loses. We now illustrate the methodology depicted in Fig. 1 by applying the Exploration phase to generate a win-win strategy for Adobe and Apple. In the Evaluation phase, we use *i** to contemplate and create new strategic options.

Figure 5a presents an extended actor model showing the *goals* of Adobe and Apple. Existing model elements are denoted by black color while new model elements are denoted by blue color. It is possible that SMEs at Adobe predicted that Apple was unlikely to greenlight either of Adobe's As-Is strategies (of translation or commingling) because each of these strategies would result in the denial of Apple's *softgoals*. Moreover, Adobe SME's probably recognized the asymmetry in the bargaining power between Apple and Adobe because Apple governed and controlled the iOS platform at its own sole discretion. Therefore, Adobe needed to generate new strategies that could help it to satisfy its own *goals* while enabling Apple to meet its objectives as well.

The Exploration phase offers five possible activities for generating new win-win strategies. These pertain to adding, removing, or changing *goals*, *dependencies*, *softgoals*, *actors*, and *tasks*. In terms of *goals*, Adobe wanted to bring support for Flash to popular mobile devices. It could have changed its *goal* to making Flash apps compatible with Android devices (not shown). With respect to *dependencies*, Adobe could have tried to change its relationship with Apple purely at the interface level. It could have paid fees to Apple to induce Apple to support its chosen option (not shown). In terms of *softgoals*, Adobe could influence Apple to modify its *softgoals.* Adobe could mount a public relations campaign to encourage Apple to support Flash (not shown).

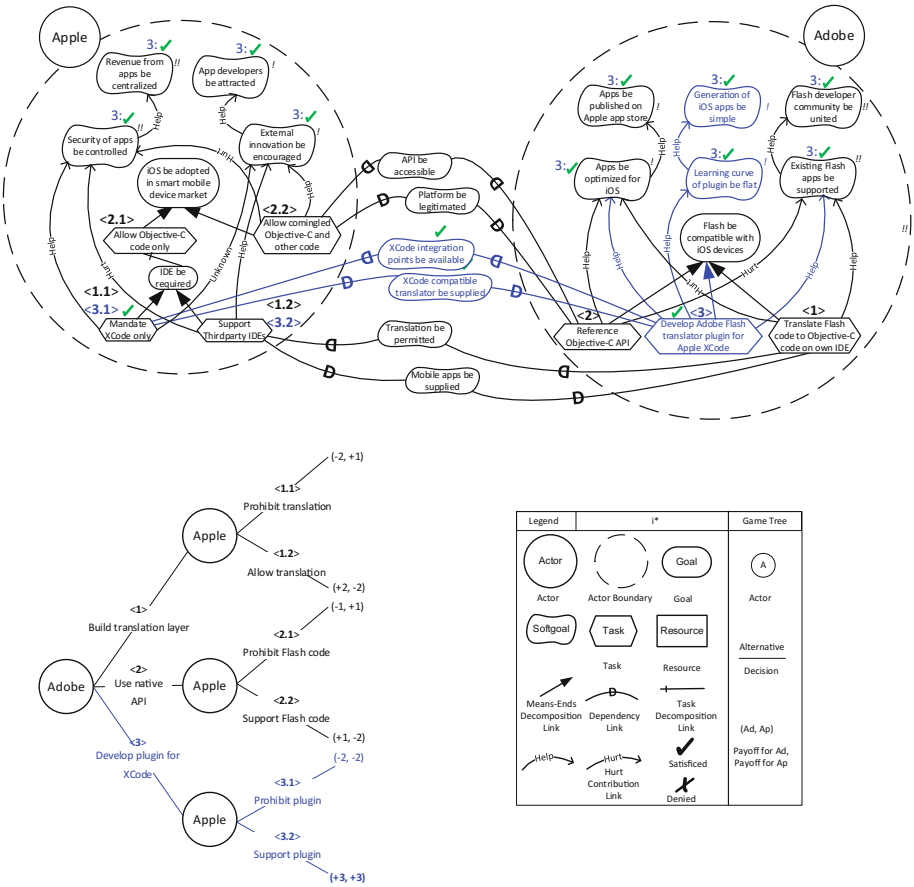

Fig. 5. (a) *i** SR model depicting To-Be actor relationships. (b) Game Tree depicting To-Be decisions and payoffs (Adapted from [12])

With respect to *actors*, Apple or Adobe were in a dyadic relationship. Adobe could have incented Apple to add support for Flash into iOS by bringing a new *actor* (e.g., its community of Flash app developers) into this relationship. Access to a large developer community that was willing to embrace iOS app development could be persuasive and compelling for Apple (not shown).

In terms of *tasks*, Adobe SMEs might have reasoned that Adobe needed to generate new alternatives in its search for a win-win strategy. Adobe SMEs likely recognized that Flash support on iOS could help Apple to satisfy its *softgoals* of "encouraging external innovation" and "attracting App developers". However, Adobe might also have understood that Apple would not support Flash on iOS if it meant that its more important *softgoals* (i.e., "Security of apps be controlled" and "Revenue from apps be centralized") were denied. Therefore, Adobe would have needed to create a new alternative that would be helpful for Apple to achieve its higher priority *softgoals*.

Starting with existing options to create new options is useful because the impact of existing options on extant intentional elements of *actors* is likely to be well understood in the Evaluation phase. As shown in Fig. 4a, translation option (<1>) was preferable to Adobe over commingling option (<2>) since the former satisfied its higher priority *softgoals* while the latter satisfied its lower priority *softgoals* (comparing <1> and <2> in Fig. 4a). However, Adobe's operationalization of the translation option (<1>) via its own IDE (Adobe Flash Builder) made it unacceptable for Apple. This is because it countermanded Apple's higher priority *softgoal* of "security of apps be controlled" and its related higher priority *softgoal* of "Revenue from apps be centralized".

However, a different implementation of the translation option might have helped Adobe and Apple to achieve their higher priority *softgoals*. For example, Adobe could have developed "Adobe Flash translator plugin for Apple XCode". Such a plugin could be embedded within XCode and could automatically inherit and apply the security policies implemented by Apple in its IDE. In such an implementation, app developers would have been able to convert Flash code into Objective-C code using XCode rather than Flash Builder. Developers of Flash apps would have had a minimal learning curve ("Learning curve of plugin be flat") which would have been limited to learning the usage of the Adobe supplied translator plugin inside XCode ("Generation of iOS apps be simple"). Apple would have been satisfied knowing that the output of this translator plugin would be Objective-C code generated inside XCode. Likewise, Adobe would have been contented knowing that its Flash apps would be supported on Apple iOS devices. Eaton et al. [25] have noted that various blogs and online news articles about Apple's service system discussed an Adobe Flash Plug-in option that was not realized. The systematic method proposed in this paper can be used to generate such a novel solution. However, it cannot replace creative thinking and deep domain knowledge but rather support and supplement it.

Figure 5b presents an extended game tree showing the payoffs for Adobe and Apple. Two decision paths at the top of this game tree are the same as those in Fig. 4b. The decision path on the bottom of this game tree reflects the new alternative that is present in Fig. 5a. This decision path is shown in blue color to differentiate it from the others. If Adobe were to select the plugin option and Apple supported it then both Adobe and Apple would have obtained payoffs of +3 each. This is because both actors could have satisfied each of their *softgoals*. Additionally, this new *task* would have unlocked additional *softgoals* for Adobe. However, if Adobe were to select the plugin option and Apple blocked it then both Adobe and Apple would have obtained payoffs of −2. This is because neither of the actors would have been able to fulfil any of their *softgoals* and would have missed out on a promising business opportunity. Therefore, this plugin option represents a win-win strategy for Adobe and Apple wherein both actors would be better off if they operationalize it as partners.

In this example, a win-win strategy was arrived at in one iteration. In the general case, one may need to go through various paths in the exploratory phase multiple times to arrive at a win-win strategy. For instance, in this example, Adobe was able to generate a new alternative ("Develop Adobe Flash translator plugin for Apple XCode") that was compatible with an element of Apple's internal intentional structure ("Mandate XCode only"). However, generation of win-win strategies in other cases may

require changes to be made to the internal intentional structures of multiple actors. Such cases will necessitate multiple iterations over different paths of this process.

Similarly, additional iterations of this process would yield other win-win strategies. For instance, in this example, Adobe could have performed additional exploration to generate other alternatives that resulted in win-win. It could have developed a translator that converted Flash code to HTML5 code since iOS supported HTML5 (not shown). Alternatively, it could have developed a translator that converted Flash code to JavaScript since iOS supported JavaScript (not shown). It is conceivable that each of these options might have led to better *payoffs* for Adobe and Apple.

5 Related Work

A number of researchers have contributed to research in these areas: (1) game-theoretic analysis of coopetition, and (2) model-based analysis of SECOs. Brandenburger and Nalebuff [3] introduced the idea of coopetition based on game theory. They also explicated facets of coopetition such as players, added value, roles, tactics, and scope [14]. Nalebuff and Brandenburger [28] defined the roles of complementors and substitutors in coopetition. Brandenburger and Stuart [29] applied cooperative game theory for strategy development. They also introduced a model of biform games to explain noncooperative-cooperative games [30]. These works do not provide a systematic method for exploring the space of strategic moves to generate new win-win strategies.

Fricker [31] developed a framework for analyzing SECO requirements using ideas from negotiation and network theories. Handoyo et al. [8] developed value chains to identify key actors and roles in SECOs. Jansen et al. [32] proposed a set of universal requirements and understandings about SECO modeling. Santos [33] developed Power Models for assessing power in SECOs.

Yu and Deng [11] were the first to use *i** strategic modeling to analyze SECOs. Pant and Yu [12] introduced a methodology for modeling strategic moves and reciprocity among actors. This methodology introduced synergistic links between Game Trees and *i** models. It was accompanied with a set of guidelines for, "instantiating an *i** SR model and its complementary game tree in a consistent manner" [12]. Key assumptions of this methodology included: (i) focal-actor orientation; (ii) distinctive preference profiles and idiosyncratic interest structures of actors; and (iii) information imperfection, incompleteness, and asymmetry [12]. The present paper extends that work by offering a systematic method for generating win-win strategies.

6 Conclusions and Future Work

This paper contributed to a line of research that links strategic modeling (*i**) with decision analysis (Game Tree). The primary contribution in this paper was in the Exploration phase of this methodology. This phase is crucial for incremental and iterative generation of win-win strategies. This phase of the methodology was explicated using a simple example predicated on minimax and maximin theorems from

Game Theory. This paper illustrated this methodology by applying it to instantiate models of SECOs under coopetition based on a published case study.

The next step in this research area is to accommodate additional facets of complexity in two-person zero-sum games within Game Trees [35]. A following step to this concerns empirical validation of this methodology in a real-life case study. Progression from validation using a published case study to validation using an empirical case study will surface the strengths and weaknesses of applying this methodology in the field. Another advancement in this research area will come from further elaboration and explication of the Exploration phase. More structured and systematic guidelines for selecting among the five steps in that phase will support practitioner efforts to use this methodology in industrial settings.

Additional areas for exploration include adding support in *i** for: (i) temporal reasoning, (ii) expressing negative dependencies, and (iii) conditional logic. The notions of time and sequence are relevant for analyzing coopetition since path dependent phenomena such as reciprocity and trust impact the moves and countermoves of actors. The depiction of negative dependencies is necessary for analyzing coopetition because the coincidental absence of dependencies and the intentional independence between actors can impel or impede coopetitive strategies. Support for conditional logic is relevant for representing cause and effect relationships such as those between the actions and responses of coopeting actors. These additions to the expressiveness of *i** can support a practitioner to more fully portray and understand the motivations behind the decisions and actions of actors in coopetitive relationships.

References

1. Chesbrough, H.: Open innovation: a new paradigm for understanding industrial innovation. In: Open Innovation: Researching a New Paradigm, pp. 0–19, 400 p. (2006)
2. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: a research agenda for software ecosystems. In: 31st International Conference on Software Engineering Companion Volume, pp. 187–190. IEEE, May 2009
3. Brandenburger, A.M., Nalebuff, B.J.: Co-opetition. Doubleday, New York (1996)
4. Duc, A.N., Cruzes, D.S., Hanssen, G.K., Snarby, T., Abrahamsson, P.: Coopetition of software firms in open source software ecosystems. In: Ojala, A., Holmström Olsson, H., Werder, K. (eds.) ICSOB 2017. LNBIP, vol. 304, pp. 146–160. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69191-6_10
5. Coutinho, E.F., Viana, D., dos Santos, R.P.: An exploratory study on the need for modeling software ecosystems: the case of SOLAR SECO. In: Proceedings of the 9th International Workshop on Modelling in Software Engineering, pp. 47–53. IEEE Press, May 2017
6. Rausch, A., Bartelt, C., Herold, S., Klus, H., Niebuhr, D.: From software systems to complex software ecosystems: model- and constraint-based engineering of ecosystems. In: Münch, J., Schmid, K. (eds.) Perspectives on the Future of Software Engineering, pp. 61–80. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37395-4_5
7. Boucharas, V., Jansen, S., Brinkkemper, S.: Formalizing software ecosystem modeling. In: Proceedings of the 1st International Workshop on Open Component Ecosystems, pp. 41–50. ACM, August 2009

8. Handoyo, E., Jansen, S., Brinkkemper, S.: Software ecosystem modeling: the value chains. In: Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems, pp. 17–24. ACM, October 2013
9. Handoyo, E.: Software ecosystem modeling. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 227–228. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39336-5_25
10. Alves, A.M., Pessoa, M., Salviano, C.F.: Towards a systemic maturity model for public software ecosystems. In: O'Connor, R.V., Rout, T., McCaffery, F., Dorling, A. (eds.) SPICE 2011. CCIS, vol. 155, pp. 145–156. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21233-8_13
11. Yu, E., Deng, S.: Understanding software ecosystems: a strategic modeling approach. In: Proceedings of the Third International Workshop on Software Ecosystems, Brussels, Belgium, pp. 65–76, June 2011
12. Pant, V., Yu, E.: Understanding strategic moves and reciprocity on software ecosystems: a strategic modeling approach. In: 9th International Workshop on Software Ecosystems (IWSECO 2017) (2017)
13. Padula, G., Dagnino, G.B.: Untangling the rise of coopetition: the intrusion of competition in a cooperative game structure. Int. Stud. Manag. Organ. **37**(2), 32–52 (2007)
14. Brandenburger, A.M., Nalebuff, B.J.: The Right Game: Use Game Theory to Shape Strategy. Harvard Business Review, pp. 57–71 (1995)
15. Dixit, A.K., Nalebuff, B.: The Art of Strategy: A Game Theorist's Guide to Success in Business & Life. WW Norton & Company, New York (2008)
16. Magdon-Ismail, M., Busch, C., Krishnamoorthy, M.S.: Cake-cutting is not a piece of cake. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 596–607. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36494-3_52
17. Barbanel, J.B., Brams, S.J., Stromquist, W.: Cutting a pie is not a piece of cake. Am. Math. Mon. **116**(6), 496–514 (2009)
18. Chen, Y., Lai, J.K., Parkes, D.C., Procaccia, A.D.: Truth, justice, and cake cutting. Games Econ. Behav. **77**(1), 284–297 (2013)
19. Deng, X., Qi, Q., Saberi, A.: Algorithmic solutions for envy-free cake cutting. Oper. Res. **60**(6), 1461–1476 (2012)
20. Aziz, H., Mackenzie, S.: A discrete and bounded envy-free cake cutting protocol for any number of agents. In: 57th Annual Symposium on Foundations of Computer Science, pp. 416–427. IEEE, October 2016
21. Dall'Aglio, M., Hill, T.P.: Maximin share and minimax envy in fair-division problems. J. Math. Anal. Appl. **281**(1), 346–361 (2003)
22. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press, Cambridge (2011)
23. Ghazawneh, A., Henfridsson, O.: Governing third-party development through platform boundary resources. In: Proceedings of the 31st International Conference of Information Systems (ICIS), St. Louis (2010)
24. Ghazawneh, A., Henfridsson, O.: Micro-strategizing in platform ecosystems: a multiple case study. In: Proceedings of the 32nd International Conference on Information Systems (ICIS) 2011, Shanghai, China (2011)
25. Eaton, B., Elaluf-Calderwood, S., Sorensen, C., Yoo, Y.: Distributed tuning of boundary resources: the case of Apple's iOS service system. MIS Q.: Manag. Inf. Syst. **39**(1), 217–243 (2015)
26. Prince, J.D.: HTML5: not just a substitute for flash. J. Electron. Resour. Med. Libr. **10**(2), 108–112 (2013)

27. Elaluf-Calderwood, S.M., Eaton, B.D., Sørensen, C., Yoo, Y.: Control as a strategy for the development of generativity in business models for mobile platforms. In: 15th International Conference on Intelligence in Next Generation Networks (ICIN), pp. 271–276. IEEE, October 2011
28. Nalebuff, B.J., Brandenburger, A.M.: Co-opetition: competitive and cooperative business strategies for the digital economy. Strategy Leadersh. **25**(6), 28–33 (1997)
29. Brandenburger, A.M., Stuart, H.W.: Value-based business strategy. J. Econ. Manag. Strategy **5**(1), 5–24 (1996)
30. Brandenburger, A., Stuart, H.: Biform games. Manage. Sci. **53**(4), 537–549 (2007)
31. Fricker, S.: Specification and analysis of requirements negotiation strategy in software ecosystems. In: Proceedings of International Workshop on Software Ecosystems (2009)
32. Jansen, S., Handoyo, E., Alves, C.: Scientists' needs in software ecosystem modeling. In: Proceedings of the International Workshop on Software Ecosystems (2015)
33. Santos, G.A.V.: A theory of power in software ecosystems formed by small-to-medium enterprises. Ph.D. thesis (2016)
34. Boudreau, K.J.: Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. Organ. Sci. **23**(5), 1409–1427 (2012)
35. Koller, D., Megiddo, N.: The complexity of two-person zero-sum games in extensive form. Games Econ. Behav. **4**(4), 528–552 (1992)