Understanding strategic moves and reciprocity on Software Ecosystems: 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
[2]Department of Computer Science, University of Toronto, Toronto, Canada
eric.yu@utoronto.ca

Abstract. Many software ecosystems comprise rival vendors that cooperate and compete with each other simultaneously. This type of relationship is termed coopetition wherein enterprises cooperate to increase collective benefits while competing to maximize their individual gains. In such a relationship, strategic moves by an actor can have significant consequences for other actors in the ecosystem. This paper proposes a model-based approach for analyzing strategic moves in software ecosystems using *i** and game trees. We offer a methodology for developing complementary *i** models and game trees. We also explicate guidelines for applying this methodology in a consistent manner. We draw upon a published case study as an illustrative example and instantiate a model based on it to assess the strengths and weaknesses of this methodology.

Keywords: Software Ecosystems. Coopetition. Reciprocity. Analysis, Game Tree, Business Modeling.

1 Introduction

Modern enterprises increasingly join ecosystems to realize benefits from synergy and complementarity between partners [1][2]. Software vendors use software ecosystems (SECO) for many purposes such as developing their apps on a common platform and taking their apps to market through a shared catalog. SECOs are operated by cloud-only vendors as well as by traditional vendors of on-premise applications. Examples of the former include Anaplan AppHub, Box Apps Marketplace, NetSuite SuiteApp, Salesforce AppExchange, and Slack App Directory. Examples of the latter include IBM Cloud Shop, Microsoft AppSource, Oracle Cloud Marketplace, SAP App Center, and VMWare Solution Exchange.

Many ecosystem participants are rivals that engage in coopetition [3]. Coopetition refers to simultaneous cooperation and competition between actors such as enterprises [4]. Each enterprise in a coopetitive relationship attempts to increase collective/common benefits through cooperation while maximizing individual/private gains via competition [5]. However, cooperation and competition are diametrical social behaviors with antipodal logics, assumptions, and hypotheses [6]. This tension [7] between cooperation and competition can create a paradox [8] for decision-makers that has the potential of undermining the stability and sustainability of the collective (i.e., SECO) as well as the individual (i.e., enterprises that are participating in a SECO).

Modeling can help decision-makers by supporting the representation and reasoning of coopetitive strategies within SECOs in a structured and systematic manner. Modeling can be used to evaluate the advantages and disadvantages as well as tradeoffs of various strategic alternatives available to participants in SECOs. Moreover, it can be used by SECO participants to compare known alternatives for coopeting as well as to generate new alternatives.

Management researchers have identified key characteristics of coopetition [9-17]. These include reciprocity [14], complementarity [15], interdependence [16], and trustworthiness [17]. Reciprocity refers to tit-for-tat (TFT) behavior in which an actor responds to a move by another actor, that it perceives to be cooperative or competitive, in a symmetrical manner. Sundali and Seale [18] note that reciprocity is a useful predictor of countermoves within coopetitive relationships.

Cygler and Sroka [19] assert that reciprocity increases the welfare in a relationship because cooperative actions by an actor are rewarded by cooperative actions of other actors in return. However, Ma [20] notes TFT is not an ideal strategy in all situations because under certain circumstances forbearance yields more favorable outcomes than retaliation. Therefore, in coopetitive relationships, it is important for decision-makers to understand when and why restraint is preferable to revenge.

The scenarios of multi-listing and multi-homing on SECOs illustrates reciprocity in action. The manager of a SECO (e.g., operator or provider) may allow competing offerings from rival software vendors (i.e., app developer) to be listed in its catalog (i.e., multi-listing). In response, an app developer may list its offerings in catalogs on multiple competing platforms (i.e., multi-homing). Alternatively, a SECO manager may allow privileged access to a section of its catalog to a specific app developer so that only that app developer can offer its apps in that category. In response, that app developer may decide to offer its apps exclusively on that SECO catalog.

In this paper, we propose a complementary approach for modeling reciprocity using *i** and game trees in a synergistic manner. Game trees elide the intentional structure of the players while *i** omits the notion of time and sequence. However, by using game trees and *i** together we overcome the individual limitations of each approach. We argue that *i** and game trees provide a stronger basis for analyzing decisions, about strategic moves and reciprocation, together rather than separately.

We discuss the need for modeling reciprocity in coopetitive relationships. We also elaborate on the concepts of strategic moves and reciprocity in coopetition. In the third section, we propose an approach for modeling strategic moves and reciprocity using *i** and game trees. In the fourth section, we draw upon a published case study as an illustrative example and instantiate a model of strategic moves and reciprocity based on it. In the fifth section, we discuss related work on modeling of SECOs. In the sixth and seventh sections, we discuss key findings from this line of research and outline future work.

2 Strategic Moves and Reciprocity in Coopetition

In the literature on coopetition, reciprocity refers to "rewarding kindness with kindness and punishing unkindness with unkindness" [21]. It facilitates cooperation among actors by serving as a guarantor of favorable treatment and a protector against injurious behavior. It "is a rather stable behavioral response by a nonnegligible fraction of the people" [22] and thus each social actor should "expect this behavior from others" [23].

Strategic reciprocity has been observed in many industrial settings [24]. However, despite its ubiquity and enduring nature as an influencer of social behavior, there are circumstances in which reciprocity can yield counter-productive decisions that lead to detrimental courses of action. Therefore, decision-makers within organizations can benefit by analyzing reciprocity using a systematic and structured approach. For instance, a decision-maker might need to evaluate whether a strategic move will lead to a mutually beneficial or unilaterally advantageous outcome for stakeholders.

Reciprocity drives actor behavior in many intra- and inter-organizational [25] as well as inter-personal [26] relationships. It serves as a guideline for action whereby one actor is assured that its beneficial/deleterious actions towards other actors will be met by a symmetrical response from those other actors. Reciprocity also impacts trust between actors because it incorporates mutuality into a relationship such that good behavior by an actor is rewarded with an increase in trust while bad behavior is punished with a decrease in trust [27]. Reciprocity can be justified in any relationship because it signifies the universal qualities of balance (symmetry) and fairness (equality).

Reciprocity has been studied extensively by researchers of economics, sociology, and psychology to explain moves and countermoves of actors in many types of social relationships. In game theory, it is a basic assumption in many sequential move games, such as gift-exchange game and ultimatum game [28]. For example, in the multi-listing scenario, if a manager decides to allow rival app developers to offer their apps in its catalog then some of those app developers may defect from that platform and switch to other ecosystems. Conversely, in the multi-homing scenario, if an app developer decides to offer its apps in catalogs on competing SECOs then a manager of one of those SECOs may exclude that app developer from its ecosystem.

This can lead to a series of reprisals that destabilize and undermine the relationship between managers and app developers on SECOs to their collective detriment. Conversely, a SECO manager or app developer may forego their impulse for short-term TFT by thoroughly evaluating the longer-term implications of reciprocal reactions. Therefore, a decision-maker might need to assess whether a reflexively reciprocal reaction will merely satisfy its short-term cravings or fulfill its longer-term strategic objectives.

The universality and timelessness of reciprocity, as a motivator of actions/reactions in strategic relationships, necessitate an approach for analyzing it in a detailed and deliberate manner. Such an approach can enable a decision-maker to take history into account and make tradeoffs between long and short-term interests. A decision-maker might need to use such an approach to reason about behaviors that entail one actor retaliating to the most recent move by another actor by taking the most self-interested immediate action, thus potentially foregoing longer-term strategic interests.

3 Case Study: Adobe Flash & Apple iOS App Store Ecosystem

The business relationship between Apple and Adobe is instructive for illustrating the impact of strategic moves and reciprocity on SECOs. Apple and Adobe were rivals because each intended to establish market superiority of its ecosystem and technology platform [35][36]. Figure 1a depicts the intentional structures of Apple and Adobe as well as their dependencies on each other via an actor-model. Figure 1b represents a game tree showing the interrelated decision spaces of these actors.

A game tree offers an approach for decision analysis when the decisions of one actor impact the decisions of other actors and vice versa. Game trees are variations of decision trees in that, while decision trees depict the decisions of a single actor, game trees portray the decisions of multiple actors [29]. They are relevant for modeling reciprocity in coopetition because the decisions of an actor can trigger TFT countermoves by other actors.

Ziegler [30] notes that "a game tree is a graphical representation of the players' possible choices (also called their action sets) at each point in time, the sequence in which these choices are made, and the payoffs resulting from any combination of choices." It supports the depiction of decisions, their sequence in terms of precedence and subsequence, as well as the payoffs associated with each decision path from root to leaf nodes. Borovska and Lazarova [31] point out that game trees can be used to "find the optimal strategy as a sequence of best possible moves of a given player taking into account possible moves of the other player up to a given depth."

Since decisions, their sequence, and payoffs are included in game trees they can be used to solve optimization problems by searching the space of alternatives and outcomes. However, for an actor to develop a game tree it is necessary for that actor to understand the intentional structures of other actors that are included in that game tree. This is required for approximating the payoffs for those actors from various decision paths. Each actor may evaluate a decision path differently based on its preferences and priorities. Therefore, any actor that wishes to reason about the perceived payoffs for other actors must recognize the unique propensities and proclivities of each of those actors.

We posit that game trees and actor modeling with *i** can be used together to achieve a deeper understanding of the decision space as well as to secure a stronger decision rationale. This is because while game trees support the depiction of payoffs they do not explicitly codify the reasons for those payoffs. While payoffs are calculated values that are based on the intentions and motivations of actors these features are only indirectly reflected within those payoffs.

However, even though the internal intentional structure of an actor cannot be expressed directly in game trees it can be represented via *i** Strategic Rationale (SR) diagrams. *i** is a socio-technical modeling language that is useful for analyzing intentional relationships between actors [32]. Its key elements are actors, goals, tasks, softgoals, and resources while its main links are for representing means-ends, decomposition, contribution, and dependency relationships. A detailed summary of *i** is offered in [33].

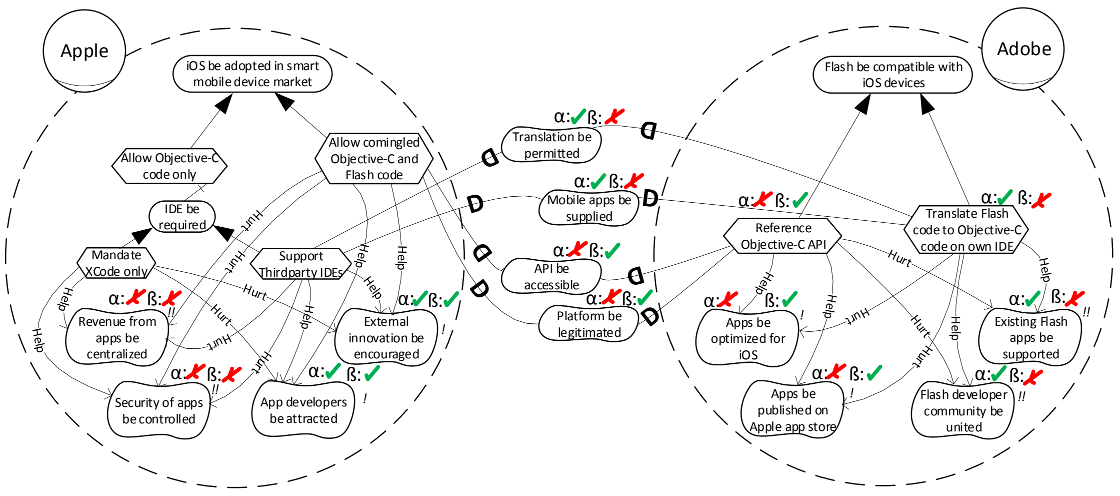
Apple
Adobe
iOS be adopted in smart mobile device market
Allow Objective-C code only
Allow comingled Objective-C and Flash code
IDE be required
Mandate XCode only
Support Thirdparty IDEs
Revenue from apps be centralized
External innovation be encouraged
Security of apps be controlled
App developers be attracted
Translation be permitted
Mobile apps be supplied
API be accessible
Platform be legitimated
Flash be compatible with iOS devices
Reference Objective-C API
Translate Flash code to Objective-C code on own IDE
Apps be optimized for iOS
Existing Flash apps be supported
Apps be published on Apple app store
Flash developer community be united
α: ✓ ß: ✗
α: ✗ ß: ✓
Help
Hurt
D
!
!!

Figure 1a. *i** SR diagram showing decision scenarios (α and ß) for Adobe and Apple

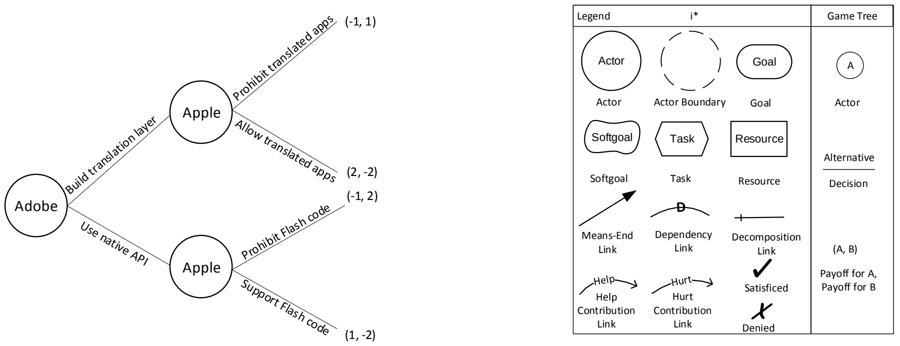

Figure 1b. Game tree of the decisions and payoffs of Adobe and Apple

Apple operated an App Store that contained a catalog of apps that were compatible with iOS devices (iPod, iPhone, and iPad). These apps were developed using a proprietary Apple programming language (Objective C) on an integrated development environment (XCode). Apple dictated strict terms and conditions for developing apps and for listing those apps on its ecosystem. For example, Apple stipulated that apps could only conduct monetary transactions with users via its Apple payment gateway. Candidate apps by third parties were also screened by Apple for compliance with its rules.

Adobe Flash was already widely adopted on the public Internet as a multimedia technology for webpages. Adobe operated its own ecosystem (Flash Gallery) that comprised of Flash-based apps. Flash was supported by many desktop web browsers via plug-ins and Adobe's intention was to bring Flash-based apps to touchscreens on mobile devices (e.g., on Apple iOS, Google Android, etc.). Third-party app developers welcomed the opportunity for expanding the reach of their existing Flash-based apps for desktop web browsers onto mobile devices such as smartphones and tablets.

In *i**, an actor is an entity that is characterized by its intentionality, autonomy, sociality, abstract/physical identity, contingent boundary, strategic reflectivity, and pursuit of rational self-interest [34]. In figure 1a, Apple and Adobe are depicted as *actors*. A *goal* is a state of affairs in the world that an actor wishes to achieve. In figure 1a, Apple has a goal titled "iOS be adopted in smart mobile device market" while Adobe has a goal "Flash be adopted on smart mobile devices".

A *task* is an alternative means for achieving an end. For example, Adobe can make Flash compatible with iOS devices by "Referencing Objective-C API" or by "Translating Flash code to Objective-C code in its own IDE". A *softgoal* is a quality objective that is considered to be satisfied or denied only from the perspective of an actor. In figure 1a, Apple has various soft-goals with one titled "Revenue from apps be centralized" and another titled "Security of apps be controlled".

A *resource* is an informational or physical entity that is necessary for fulfilling a task. *i** also supports the depiction of the relative *priority* of an element for an actor. For example, Adobe treats "Existing Flash apps be supported" as a more important softgoal with a "*!!*" priority than "Apps be optimized for iOS", which is a softgoal with a relatively lower priority of a "!".

Various aspects of *i** make it suitable for analyzing strategic moves including its support for *means-ends* reasoning, *dependency* link evaluation, *contribution* analysis, *label propagation*, and *task-decomposition*. A *means-end* link shows the relationship between a goal and the tasks that serve as alternatives for achieving that goal. A *dependency* link connects an actor that depends (i.e., depender) to another actor on which the depender depends (i.e., dependee) for something (i.e., dependum). A *contribution* link shows the impact (e.g., help, hurt) of alternatives on softgoals.

The cascading effect of a lower level element on higher level elements that are impacted directly or indirectly is shown via *label propagation*. This allows bottom up analysis to be conducted wherein the effect of an operationalizing element can be traced upwards to strategic elements. A *decomposition* link shows the relationship between a task and its constituent parts such as sub-goals, sub-tasks, sub-softgoals, and resources. The completion of any task leads to the attainment of its corresponding goal. Each of the components of a task need to be completed for that task to be accomplished.

Adobe was the first-mover in this case, and is shown as the first decision-maker in figure 1b, since it decided to support Flash-based apps on Apple iOS devices. Adobe faced an important question regarding which of the two existing options was preferable for bringing Flash-based apps to Apple iOS devices? As shown in figure 1a, there were two main approaches for making Flash-based apps compatible with iOS devices. These approaches are represented as alternate means for achieving an end.

In one approach Flash code could be made interoperable with native iOS application programming interfaces (APIs) (i.e., comingling option). In a different approach, an Adobe IDE for Flash could serve as a translation tool to convert Flash code to Objective-C code (i.e., translation option). Each option represented different trade-offs between Adobe's various quality objectives and impacted Apple's goals differently. These are represented in figure 1a via different satisfaction or denial labels on goals and dependencies. These alternatives are labeled as α (translation) or ß (comingling) to differentiate label values that are propagated to other model elements.

The translation option (α) was preferable for Adobe over the comingling option (ß) because a drawback of the comingling option was that many extant Flash-based apps would need to be fitted with Objective-C code by developers. This would limit the reusability of Flash-based apps across platforms. It could also fragment the developer community into groups for iOS oriented developers and non-iOS oriented developers. However, the translation option also suffered from a drawback because Adobe would need to maintain stability of translated apps over time. It would need to guarantee compatibility between Flash-based apps that were translated to Objective-C apps using its IDE on different iOS versions. This could be a costly endeavor for Adobe.

Apple was the second-mover in this case since it needed to respond to Adobe's move (i.e., its decision of α or ß). Apple operated its iOS App Store unilaterally and needed to decide whether to respond to Adobe's move in a favorable or hostile manner? It could act favorably by allowing translated apps or comingling of Objective-C and Flash code. Alternatively, it could act hostilely by rejecting translated apps or disallowing mixing of Objective-C and Flash code. Apple depended on Adobe for obtaining widely used apps for its catalog or for its iOS platform to be legitimated by supporting an established technology (i.e., Flash). However, it had alternatives for each of these dependencies (not shown in figure 1a). These included adding apps from other developers (i.e., substitutes for Flash-oriented developers) and supporting HTML5 (i.e., rival of Flash)

If Adobe selected the translation option then Apple could prohibit or allow translated apps. As Figure 1b shows, allowing translated apps was not an advantageous option for Apple because it meant that developers could build iOS apps without using its IDE (XCode). This meant developers could bypass rules and regulations that Apple had baked into in its IDE for enforcing security in third-party iOS apps and its App Store. Figure 1b depicts the relative payoffs for Adobe that are calculated by considering its goal model from Figure 1a. If Adobe selected the translation option and Apple allowed translated apps then Adobe would earn a payoff of 2 and Apple would earn a payoff of -2. However, if Apple prohibited translated apps then Adobe would earn a payoff of -1 (from wasted effort in building a translator) and Apple would earn a payoff of 1.

If Adobe selected the API option then Apple could permit or block the execution of translated apps on iOS. Permitting the comingling of code written in multiple programming languages was not beneficial for Apple because app developers could obscure functionality coded in Flash within their iOS apps from Apple. For example, they could hide in-app monetization from Apple by developing it in Flash so monetary transactions could be conducted on third-party payment gateways. Figure 1b shows that if Adobe selected the comingling option and Apple supported code mixing then Adobe would earn a payoff of 1 and Apple would earn a payoff of -2. However, if Apple disallowed code mixing then Adobe would earn a payoff of -1 (from wasted effort in adding Objective-C support to Flash IDE) and Apple would earn a payoff of 2.

Reasoning based on figures 2a and 2b suggests that, given the existing configuration of alternatives, objectives and dependencies, it was disadvantageous for Apple to allow Flash-based apps to run on iOS devices regardless of Adobe's choice (translation via third-party IDE or code comingling of Objective-C and Flash). Apple intended for its ecosystem to grow (e.g., by third-party developers contributing apps) and be legitimated (e.g., by supporting established technologies such as Flash). However, Apple also intended to avoid exposing itself to vulnerabilities that could accompany support for Flash-based apps on iOS devices that were translated using third-party IDEs or developed using programming languages that could obscure app code from Apple staff.

In the case study, Adobe moved by selecting the translation option (α) and Apple swiftly retaliated by prohibiting translated apps. This blocked Flash-based apps that were translated into Objective-C apps using third-party IDEs from running on iOS devices. This case shows reciprocation in action because Adobe's move was harmful for Apple and Apple responded by moving in a way that was damaging for Adobe.

This logic leads to the inescapable conclusion that Adobe should not have attempted to bring Flash-based apps to iOS devices using the existing alternatives that it was considering. This is because the alternatives that it was contemplating would have resulted in a *win-lose* situation where it would have gained while Apple would have lost. Rather, Adobe should have generated new alternatives for creating a *win-win* situation for itself and for Apple. An example of a new alternative could be for Adobe to develop a Flash translator plugin for XCode (not shown). Such a plugin could allow importation of Flash code into XCode and be translated into Objective-C code in XCode. This could have allowed Adobe and Apple to satisfy their objectives and avoid TFT actions.

4 A Methodology for Understanding Strategic Moves and Reciprocity on Software Ecosystems with *i** & Game Trees

We propose a methodology for co-developing complementary *i** models and game trees. This methodology is presented in Figure 2. The artifacts that result from this methodology can offer a deeper understanding of the relevant decision domain as well as the problem and solution spaces. This methodology supports the assessment of *win-win* strategies to decide whether existing solutions are satisfactory or if new solutions must be generated and considered.

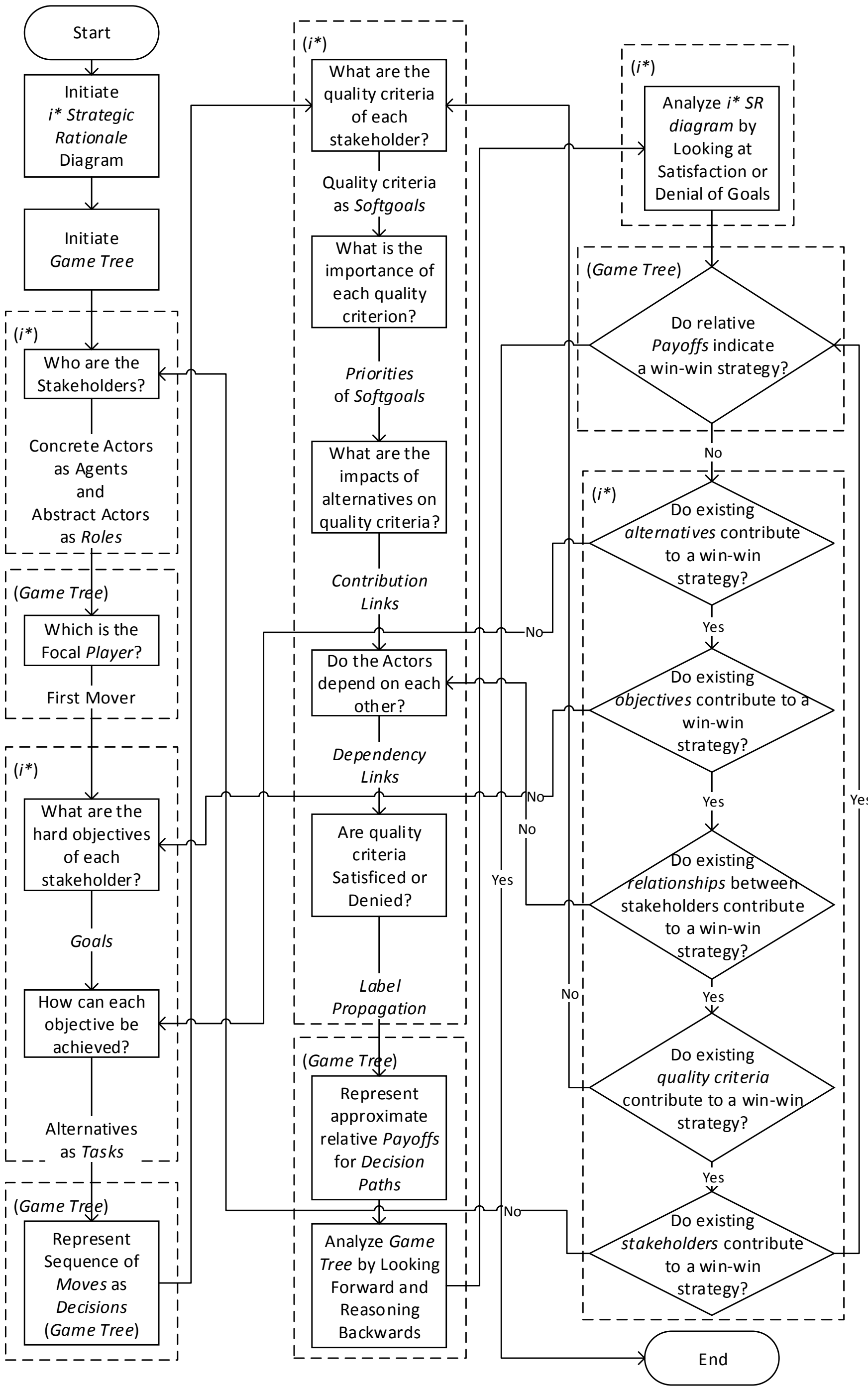
Start
Initiate i* Strategic Rationale Diagram
Initiate Game Tree
(i*)
Who are the Stakeholders?
Concrete Actors as Agents and Abstract Actors as Roles
(Game Tree)
Which is the Focal Player?
First Mover
(i*)
What are the hard objectives of each stakeholder?
Goals
How can each objective be achieved?
Alternatives as Tasks
(Game Tree)
Represent Sequence of Moves as Decisions (Game Tree)
(i*)
What are the quality criteria of each stakeholder?
Quality criteria as Softgoals
What is the importance of each quality criterion?
Priorities of Softgoals
What are the impacts of alternatives on quality criteria?
Contribution Links
Do the Actors depend on each other?
Dependency Links
Are quality criteria Satisficed or Denied?
Label Propagation
(Game Tree)
Represent approximate relative Payoffs for Decision Paths
Analyze Game Tree by Looking Forward and Reasoning Backwards
(i*)
Analyze i* SR diagram by Looking at Satisfaction or Denial of Goals
(Game Tree)
Do relative Payoffs indicate a win-win strategy?
No
(i*)
Do existing alternatives contribute to a win-win strategy?
Yes
Do existing objectives contribute to a win-win strategy?
Yes
Do existing relationships between stakeholders contribute to a win-win strategy?
Yes
Do existing quality criteria contribute to a win-win strategy?
Yes
Do existing stakeholders contribute to a win-win strategy?
No
No
No
No
No
Yes
Yes
End

Figure 2. Methodology for co-developing complementary *i** SR diagram and Game Tree

The *i** model and game tree are co-developed from the perspective of a focal actor that is contemplating a strategic move that might result in a TFT countermove. In Figure 2, each solid box represents an action step while each arrow represents a connection between action or decision steps. Dashed boxes represent groupings of solid boxes pertaining to a modeling technique. Each action step involving elicitation has an outbound arrow that is used to denote an addition to the model resulting from that elicitation step. The following guidelines are useful for instantiating an *i** SR model and its complementary game tree in a consistent manner:

1. A focal actor is unlikely to be fully knowledgeable about the true intentional structure of its counter-parties. Therefore, goal models of counter-parties are based on assumptions on the part of the focal actor that are informed by various sources. These can include past behavior and public comments of those counter-parties as well as market research and competitive intelligence about them.

2. The first decision-maker in the game tree is the focal actor. This actor is choosing among the alternatives that are available to it for initiating a strategic move that might trigger a TFT countermove. Subsequent decisions are interleaved among the counter-parties and the focal actor based on their relational context. In a simplistic game tree, decisions could be interleaved among actors in a round robin arrangement of turns.

3. The relative payoffs to each actor are based on the assumptions of the focal actor. These numbers serve a purely comparative purpose and their unit of measure is irrelevant. It is only assumed that each actor prefers a higher or positive outcome to a lower or negative outcome wherein 2 is preferable to 1, 1 is preferable 0, 0 is preferable to -1, -1 is preferable to -2, etc.

4. Each unique decision path from the root node to a leaf node on the game tree is regarded as a strategy. Payoffs denote the value gained or value lost for each player based on a decision path. Each player independently decides whether a particular decision path is advantageous or disadvantageous for it based on the value that it can expect to gain or lose.

5. A *win-win* strategy refers to a *positive-sum* strategy in which each player gains. This strategy is likely to be accepted by each of the players. When multiple win-win strategies are available then the strategy with the highest payoffs for all players is likely to result in a stable equilibrium as each player would be maximally better off with it. A *win-lose* strategy refers to a *zero-sum* situation in which some players gain at the expense of other players. A *lose-lose* strategy refers to a *negative-sum* situation in which all players lose. Both win-lose and lose-lose strategies are unlikely to lead to a stable equilibrium as some or none of the players might be willing to cooperate.

6. Payoffs for a player in the game tree represents the idiosyncratic preference structure and distinctive intentionality of that actor as represented in the *i** SR model. Payoffs can change when the objectives of that actor or the satisfaction of those objectives change. Relative payoffs are approximated based on the priorities associated with various objectives as well as their satisfaction or denial.

5 Related Work

Two related streams of SECO research pertain to the focal subject. These are: (1) foundations of SECOs and (2) model-based analysis of SECOs. Researchers have explored foundational aspects of SECOs to better explain these socio-technical phenomena. Bosch and Bosch-Sijtsema [37] trace the evolution of SECOs and their development within the broader software industry. Popp [38] classifies business models of software companies and explains key characteristics. Weiblen et al. [39] identify marketplace business models for managing SECOs. de Andrade et al. [40] investigate performance measurements for assessing the health and sustainability of SECOs. Idu et al. [41] explore categories of apps within the iOS SECO to discern the motivations and strategies of app developers. Such research about the foundational aspects of SECOs have helped to establish a consistent and coherent intellectual basis for the study of SECOs.

Researchers have also applied modeling techniques for analyzing SECOs. Boucharas et al. [42] propose a process for establishing a standard for formal modeling of SECOs which is useful for communicating, theorizing, and anticipating developments on SECOs. Fricker [43] applies concepts from negotiation and network theory to develop a framework for analyzing SECO requirements. Sadi and Yu [44] identify a set of analytical as well as descriptive requirements for SECOs and analyze the expressiveness as well as analytic capabilities of various modeling languages for supporting those requirements. Christensen et al. [45] apply the Business Model Canvas to study the business logic underlying SECO architectures. Pettersson et al. [46] apply Software Process Engineering Meta-model to study domain specific SECO. Such research that applies established techniques for representation and reasoning to study SECOs has yielded rigorous and defensible results.

Additionally, Gans et al. [47] propose a TCD (Trust–Confidence–Distrust) approach for continuous requirements management in inter-organizational networks. Their approach considers path dependency, history, and strategic dependency between actors during analysis [47]. Our work is synergistic to this approach which "combines the structural analysis of strategic dependencies and rationales, with the interaction between planning, tracing, and communicative action" [47].

6 Discussion

In this paper, we noted that SECOs comprise participants that are engaged in coopetitive relationships with each other. Within such inter-organizational relationships the decisions of an actor are supported or constrained by alliance- and network-centric aspects (e.g., the actions of other actors). We discussed that reciprocity is a common behavioral response by humans and is prevalent in many inter-organizational relationships. However, while reciprocation might be a suitable behavior in certain scenarios there are many circumstances when it is not appropriate. In Section 2, we noted certain limitations and drawbacks of reflexive reciprocation and autonomic retaliation by enterprises in coopetitive relationships.

In Section 3, we showed that, *i** and game trees can be used together to express the logic behind the moves and countermoves of two enterprises in a coopetitive relationship related to their SECOs. In section 4, we proposed a methodology for synergistically co-developing an *i** SR diagram and a game tree. These complementary artefacts are useful for comparing various strategic alternatives available for participants in a SECO.

Figure 1a demonstrated the characteristics of *i** SR diagrams that are relevant for modeling strategic moves and reciprocity. Alternatives for achieving objectives are represented as tasks linked to a goal (e.g., Apple can achieve its objective of driving adoption of iOS devices in the smart mobile device market either by allowing iOS apps to be coded only in Objective-C or with Ojective-C and Flash). Relationships between actors are shown as dependencies (e.g., Apple depends on Adobe to supply mobile apps to its App Store while Adobe depends on Apple for access to its API).

Contribution by elements to each other are portrayed via help and hurt links (e.g., Adobe can reference the Objective-C API and this can satisfy its objective of optimizing its apps for iOS devices however this can lead to the denial of its objective of existing Flash apps be supported). Label propagation is used to show the effect of choices by Adobe on the available choices for Apple through a cascading satisfaction or denial of relational elements (e.g., Adobe translating Flash code to Objective-C code on own IDE will lead to the denial of high priority objectives of Apple which are to centralize revenue from apps and control security of apps). Task decomposition is used to show a hierarchy of objectives and alternatives (e.g., Apple can allow Objective-C code only but do so by mandating XCode as the only supported IDE or by supporting third-party IDEs as well).

In section 3, we noted that *i** is useful for performing evaluative as well as generative analysis. *i** provides representation support for exploring and elaborating on the problem understanding as well as the solution space. *i** models support the evaluation of existing problems and solutions that are already represented as well as the exploration of new problems and solutions that need to be generated. This is in contrast to typical search algorithms on game trees that assume that the problem space is fully known, and that the solution space is given and fixed.

7 Conclusion and Future Work

This paper presents novel work because it proposes a modeling-based approach for analyzing strategic moves and reciprocity in coopetitive relationships on SECOs. We posit that *i** is well suited for analyzing and solving optimization problems on game trees because it supports the analysis of tradeoffs between alternatives in terms of quality objectives. *i** supports the depiction of softgoals that can be used to compare the impact of each alternative on an actor's objectives. It also supports the representation of dependencies among actors that serve as sources of opportunities/vulnerabilities thereby supporting/constraining different courses of action between them.

A limitation of *i** is that it lacks support for representing negative dependencies. *i** does not support the discrimination between the omission of dependency (i.e., dependency is not depicted in an *i** diagram) and the lack of dependency (i.e., no dependency exists between actors). Modeling reciprocity necessitates the means for representing lack of dependency because this can have an impact on the decision-making process of an actor. Lack of dependency on another actor may give more flexibility to an actor while absence of a dependency may reflect a choice by the modeler to elide certain model elements. Extending *i** to express the distinction between absence of a dependency and a lack of a dependency will aid the ability to develop more thorough models.

Validating our approach of jointly using *i** and game trees further is a part of future work as is representing and reasoning about the temporal dimension of reciprocity. The approach that is presented in this paper can be tested by applying it to additional published case studies. It can also be tried in the industry through empirical case studies in organizational settings. It can also be critiqued by enterprise modeling practitioners that are familiar with the use of decision-support tools such as game trees.

References

1. Gnyawali, D. R., & Park, B. J. R. (2011). Co-opetition between giants: Collaboration with competitors for technological innovation. *Research Policy*, 40(5), 650-663.
2. Brandenburger, A. M., & Nalebuff, B. J. (1996). *Co-opetition*. New York: Doubleday.
3. Harbison, J.R., & Pekar, P.P. (1998). *Smart Alliances: A Practical Guide To Repeatable Success*. San Francisco, CA: Jossey-Bass.
4. Brandenburger, A. M., & Nalebuff, B. J. (1995). The right game: Use game theory to shape strategy. *Harvard Business Review*, 73(4), 57-71.
5. Nalebuff, B. J., & Brandenburger, A. M. (1997). Co-opetition: Competitive and cooperative business strategies for the digital economy. *Strategy & Leadership*, 25(6), 28-33.
6. Bengtsson, M., & Kock, S. (2000). "Coopetition" in business Networks—to cooperate and compete simultaneously. *Industrial Marketing Management*, 29(5), 411-426.
7. Raza-Ullah, T., Bengtsson, M., & Kock, S. (2014). The coopetition paradox and tension in coopetition at multiple levels. *Industrial Marketing Management*, 43(2), 189-198.
8. Gnyawali, D. R., Madhavan, R., He, J., & Bengtsson, M. (2016). The competition–cooperation paradox in inter-firm relationships: A conceptual framework. *Industrial Marketing Management*, 53, 7-18.
9. Gnyawali D.R., & Park, B. (2009). Co-opetition and technological innovation in small and medium-sized enterprises: A multilevel conceptual model. *Journal of Small Business Management* 47(3):308-330.
10. Zineldin, M. (2004). Co-opetition: the organisation of the future. *Marketing Intelligence & Planning*, 22(7), 780-790.
11. Chin, K.S., Chan, B.L., & Lam, P.K. (2008). Identifying and prioritizing critical success factors for coopetition strategy. *Industrial Management & Data Systems*, 108(4), 437-454.
12. Bonel, E., Pellizzari, P., & Rocco, E. (2008). Coopetition and Complementarities: Modeling Coopetition Strategy and Its Risks at an Individual Partner Level. *Management Research Journal of the Iberoamerican Academy of Management*, 6(3), 189-205.
13. Bengtsson, M., Eriksson, J., & Wincent, J. (2010). Co-opetition dynamics-an outline for further inquiry. *Competitiveness Review: An International Business Journal*, 20(2), 194-214.

14. Rossi A, Warglien M (2000) An experimental investigation of fairness and reciprocity as determinants of intraorganizational coopetition. In: Dagnino GB, Rocco E (Eds.), *Coopetition Strategy: Theory, Experiments and Cases*. Routledge, New York, pp. 258–273.
15. Tee, R., & Gawer, A. (2009). Industry architecture as a determinant of successful platform strategies: A case study of the i- mode mobile Internet service. *European Management Review*, 6(4), 217-232.
16. Luo, Y. (2005). Toward coopetition within a multinational enterprise: a perspective from foreign subsidiaries. *Journal of World Business*, 40(1), 71-90.
17. Bouncken, R.B., & Fredrich, V. (2012). Coopetition: performance implications and management antecedents. *International Journal of Innovation Management*, 16(5), 1-28.
18. Sundali, J. A., & Seale, D. A. (2002). Is reciprocity necessary for coopetition? Some experimental evidence. *Journal of Behavioral and Applied Management*, 4(1), 68-87.
19. Cygler, J., & Sroka, W. (2016). The boundaries of coopetition: A case study of polish companies operating in the high-tech sector. In: *Economic Development and Entrepreneurship in Transition Economies* (pp. 253-269). Springer.
20. Ma, H. (1998). Mutual forbearance in international business. *Journal of International Management*, 4(2), 129-147.
21. Ashraf, N., Bohnet, I., & Piankov, N. (2006). Decomposing trust and trustworthiness. *Experimental Economics*, *9*(3), 193-208.
22. Fehr, E., & Gächter, S. (2000). Fairness and retaliation: The economics of reciprocity. *The Journal of Economic Perspectives*, *14*(3), 159-181.
23. Sobel, J. (2005). Interdependent preferences and reciprocity. *Journal of Economic Literature*, *43*(2), 392-436.
24. Krämer, A., Jung, M., & Burgartz, T. (2016). A small step from price competition to price war: Understanding causes, effects and possible countermeasures. *International Business Research*, 9(3), 1.
25. Hu, Y., & Korneliussen, T. (1997). The effects of personal ties and reciprocity on the performance of small firms in horizontal strategic alliances. *Scandinavian Journal of Management*, 13(2), 159-173.
26. Buunk, B. P., & Schaufeli, W. B. (1999). Reciprocity in interpersonal relationships: An evolutionary perspective on its importance for health and well-being. *European Review Of Social Psychology*, 10(1), 259-291.
27. Altmann, S., Dohmen, T., & Wibral, M. (2008). Do the reciprocal trust less?. *Economics Letters*, 99(3), 454-457.
28. Falk, A., & Fischbacher, U. (2006). A theory of reciprocity. *Games and Economic Behavior*, 54(2), 293-315.
29. Dixit, A. K., & Nalebuff, B. (2008). *The Art of Strategy: A Game Theorist's Guide To Success In Business & Life*. WW Norton & Company.
30. Ziegler, A., 2004. *A Game Theory Analysis of Options: Corporate Finance and Financial Intermediation In Continuous Time*. Springer.
31. Borovska, P., & Lazarova, M. (2007). Efficiency of parallel minimax algorithm for game tree search. In: *Proceedings of the 2007 International Conference on Computer Systems and Technology*. ACM.
32. Yu, E. (2001). Agent-oriented modelling: software versus the world. In: *International Workshop on Agent-Oriented Software Engineering* (pp. 206-225). Springer.

33. Yu, E. S., & Mylopoulos, J. (1994). Understanding" why" in software process modelling, analysis, and design. In: *Proceedings of 16th International Conference on Software Engineering* (ICSE-16) (pp. 159-168). IEEE.
34. Yu, E. (2001). Agent orientation as a modelling paradigm. *Wirtschaftsinformatik*, 43(2), 123-132.
35. Ghazawneh A., Henfridsson, O. 2010. Governing third-party development through platform boundary resources, in: *Proceedings of the 31st International Conference of Information Systems* (ICIS) 2010, St. Louis, USA.
36. Ghazawneh A., Henfridsson, O. 2011. Micro-strategizing in platform ecosystems: a multiple case study, in: *Proceedings of the 32nd International Conference on Information Systems* (ICIS) 2011, Shanghai, China.
37. Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1), 67-76.
38. Popp, K. (2011). Software industry business models. *IEEE software*, 28(4), 26.
39. Weiblen, T., Giessmann, A., Bonakdar, A., & Eisert, U. (2012). Leveraging the software ecosystem: Towards a business model framework for marketplaces. In: *Proceedings of the International Conference on Data Communication Networking, eBusiness and Optical Communication Systems* (DCNET, ICE-B and OP-TICS 2012); SciTePress. 187–193.
40. de Andrade, R. V. L., Alves, C. F., & Valença, G. (2015). An Analysis of Dynamic Strategies during the Lifecycle of Software Ecosystems: The DS-SECO Model. In: *Brazilian Congress on Software: Theory and Practice* (WDES), Belo Horizonte, Brazil 2015, pp. 57–64.
41. Idu, A., van de Zande, T., & Jansen, S. (2011, November). Multi-homing in the apple ecosystem: why and how developers target multiple apple app stores. In: *Proceedings of the International Conference on Management of Emergent Digital EcoSystems* (pp. 122-128). ACM.
42. Boucharas, V., Jansen, S., & Brinkkemper, S. (2009, August). Formalizing software ecosystem modeling. In: *Proceedings of the 1st international workshop on Open component ecosystems* (pp. 41-50). ACM.
43. Fricker, S. (2009). Specification and Analysis of Requirements Negotiation Strategy in Software Ecosystems. In: *Proceedings of International Workshop on Software Ecosystems*.
44. Sadi, M. H., & Yu, E. (2015). Designing Software Ecosystems: How Can Modeling Techniques Help? In: *Enterprise, Business-Process and Information Systems Modeling* (360-375).
45. Christensen, H. B., Hansen, K. M., Kyng, M., & Manikas, K. (2014). Analysis and design of software ecosystem architectures–Towards the 4S telemedicine ecosystem. *Information and Software Technology*, 56(11), 1476-1492.
46. Pettersson, O., Svensson, M., Gil, D., Andersson, J., & Milrad, M. (2010, August). On the role of software process modeling in software ecosystem design. In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume* (pp. 103-110). ACM.
47. Gans, G., Jarke, M., Kethers, S., & Lakemeyer, G. (2003). Continuous requirements management for organisation networks: a (dis) trust-based approach. *Requirements Engineering*, 8(1), 4-22.