# Applications of Game Theory in Software Engineering

S.W.H.M.S.P Herath, U.A.R.R.Gunarathna, H.A.C Hettiarachchi, H.M.A.B Herath, S.K.K Wickramanayake
Department of Computer science and engineering, University of Moratuwa

**Abstract** — *Turning a new chapter in game theory some of its applications to the field of Software Engineering are explored recently. Game theory addresses strategic problems. There are many aspects in software development process which could be analyzed using game theory. This is a discussion and a research of how game and why theory is applied. Many real life situations including situations arise in software engineering process can be abstracted into prisoners' dilemma situations. Game theory principal usage in software engineering, technical and non technical aspects of software engineering, project management and avoiding software development failures are discussed in game theory perspective.*

*Maintenance is another major aspect of Software Engineering. Application of strategies of game theory in software maintenance would be beneficial. In that way game theory can be used for the benefit of the software engineering processes because meetings are ubiquitous in software engineering projects.*

**Index terms — Game theory, software development**

## I. Introduction

Game theory provides a mathematical model to understand, analyze, structure and describe strategic situations in which actions of multiple agents are involved. Here agents would be individuals, groups, firms, or any combination of these. Agents could have goals assigned to themselves on which their decision to pick an action depends. The formal definition lays out the players, their preferences, their information, and the strategic actions available to them, and how these influence the outcome [1].

It is important to identify the fact that actions of agents are interdependent. The outcome of an action taken by an individual concerns not only him but also the other agents in the scenario. On the other hand when one agent chooses his/her action he/she has to consider not only his but also others' state. The bottom line is that the success (achieving goal) of each agent depends on the behavior of the other agents.

Game theory is a powerful and very interesting study area of situations where attempts to capture or decide the behavior of other agents are involved to make decisions. It is obvious that most strategic games such chess has a mapping with the above description. Therefore game theory can be used to maximize the gain of such situations. Why game theory has become more interesting is, it is applicable for much broader domain. Game theory is highly used in the fields of economics and politics where decisions to be made are highly depend on what others do.

Turning a new chapter in game theory some of its applications to the field of Software Engineering are explored recently. Trade-offs between different options is a usual consideration in Software Engineering; particularly there is a trade among some non-functional requirements of a system. Game theory addresses such strategic problems. As there are many aspect in software development process which could be analyzed using game theory. Some of them are project management, costing, architecture designing and etc. This paper discusses what are the aspects that can be analyzed using game theory in order to produce a better software solution, to have better development process, and to maintain better management within the process. And also how game theory can be applied and why.

## II. Background: THE PRISONER'S DILEMMA

[1], [3] Any explanation on the concepts of the Game theory is done with respect to some classical example games. They describe some abstract situations where individuals' success depends on others' decisions and actions. It is necessary to have knowledge on these classic examples before examine any real life applications of game theory.

The prisoners' dilemma is one such example where almost any writing on Game theory refers to. Following are the information that set up prisoners' dilemma.

Police catches two convicts (who have no acquaintances between them) of a crime and imprisons them in 'separate' rooms. But the police have no evidence against any of them. So they separately ask each of the prisoners to testify against each other. Now any of the following situations could occur.

Case 1: One testifies against the other and the other doesn't.

Case 2: Both testify against each other.

Case 3: Both refuse to testify.

Consequences of the decisions of each prisoner are as follows. If refusing to testify is termed as 'co-operating' and testifying as 'defecting',

Case 1: the one who cooperate is sentenced to 10 years imprisonment and the one who defects immediately goes free.

Case 2: both will be sentenced for 3 years imprisonment. The punishment is so reduced because they both gave evidence.

Case 3: both will be released after 3 months due to lack of evidence.

As no communication is allowed between the prisoners they have no idea about the decision of the other. And observe the success (Imprisonment is a failure.) of one person depends on the decision of others. An optimistic person will cooperate assuming that the other will also cooperate. A pessimistic person will defect not expecting the other to cooperate with him. Note that the rational decision will also be to defect because that way your average sentence will be 1.5 years imprisonment. But if they both have cooperated blindly they end up in a win-win situation. The most notable issue here is that your success or failure depends on the decision of a total stranger to you.

Game theory consists of many such examples and strategies to be used in such situations. Many real life situations including situations arise in software engineering process can be abstracted into prisoners' dilemma situations. So the concepts of the Game theory can be applied to them. On the sections that follow it will be discussed how this is done.

### *A. Game Theory Principal usage in Software Engineering*

"The central to game theory is the strategies, payoffs and rational attitude of the players. Software engineering is also a human oriented endeavor and therefore all three central aspects of game theory are relevant to it" [4] pp6.

### *B. Non technical Aspects of Software Engineering*

#### *1) Costing and Bidding*

[5]Costing and bidding is one of the most essential activities during the establishment of a software project.

Let's look at Bidding in another perspective: a game which has peculiar characteristics of a game with imperfect information and infinite game.

- Game with imperfect information: when bidders place their bids they never get a chance to know the competitors' bid
- Infinite game: bidders never know how long the customer continues to ask them to bid for a new software project, so the players would not know when the game will end

In order to model the bidding behavior in Game Theory, certain behaviors should be identified.

- The budget constraints and limitations directly affect the behavior of bidders
- Each bid is an offer to do a piece of work for a particular price. This particular price is a result of the addition of cost and profit (Price = Cost + Profit). The cost is derived from the costing activity and the profit is the desirable amount that the bidder would like to earn from performing the project

Some other effects that can be recognized in conventional bidding process will be expressed in the following paragraph.

- Berk, Hughson and Vandezande investigate the television game show "The Price Is Right".

*"In this game, four opponents sequentially guess a retail price of durable goods without going over the retail price. Their main conclusions are that the fourth player has the biggest chance to win and that learning during the show reduces the frequency of strategic errors. This happens when the game is sequential like normal bidding process. But this can be changed using sealed bid auctions"* [6] pp18.

- Budget constraints the bidder, therefore they take risks when they are bidding. But it would be extremely difficult to recover if there is at least one bidder who does not take risks.

Above facts show us, some concepts of the Game Theory that enable to evaluate bidding process in software engineering. Then it can be used to make rational decisions.

#### *2) Project Management*

In a software development process, project management plays the most important role: make sure each and every stakeholder is happy about the project. This is not an easy task. It becomes worse especially when the development process becomes an agile approach, because project management would ask for inappropriate planning details even though allowing user requirements to be changed is a key characteristic of the development method. In such a situation the project management would get a bad reputation on the

development team. Sometimes this may lead even to violate the principles of agile development methodology.

"Leading Answers" [7] website presents a new perspective of the project management in terms of game theory. In this approach the project is considered as a cooperative game and members of the development team are players. Team members should collaborate to be successful. The result of the project is important but the team members should be looked after and set them to the future success also. In this point of view project management tasks can be mapped into team game approach which will help the project management to be a supporting body instead of an obstacle. Through this kind of approach project management can promote many opportunities, but the ability of mapping depends on various conditions. However with a simple shift of view point, a huge change can be achieved.

*3) Avoiding software Development Failures*

It is a harsh truth that only 35% of the software development projects become a success all round the world. The other 65% is at either partial failure or total failure. [8] This fact is affected by many reasons. Poor requirement specifications, inappropriate development methodology, wrong tool selection, poor communication among stakeholders and etc can be identified as some of those reasons. Even though it is not significant all of these causes of failures are based on an assumption that every stakeholder work towards the same goal. The reality is different. Each and every individual in the development team has his/her own goals while in general every software project is having two major goals: achieve a successful software product and conduct a successful software development and maintenance process [9 ] pp1. If those two kinds of goals conflict with each other, it becomes another major reason to make the project a failure.

Then again thinking a software development as a non-cooperative game let to identify these kinds of hidden causes of failure and can be used to even avoid them. In this context SD is considered as a game and all the participants as players having characteristics individualism, rationality, and mutual interdependence with other participants. Players can be categorized as management, customers and developers those who have different payoffs and strategies. Conflicts between different strategies of these players are addressed by using Nash Equilibrium [1].

*4) Maintenance Services*

Maintenance is another major aspect of Software Engineering. Application of strategies of game theory in software maintenance would be beneficial. According to the paper "Incentive Compatible Mechanisms for Group Ticket Allocation in Software Maintenance Services" [10] it says "A trouble ticket" (or synonymously a ticket) is a software problem as reported by a customer to be analyzed and fixed by a team of maintenance engineers. The problem ticket can come to the organization through different interfaces such as web interface system, call centers, emails etc. The ticket received through any such interface will then be channelized to a lead. The lead in turn takes the responsibility to allocate the ticket to one of the reporting engineers. The complexity of the reported problem actually propagates from the bottom layer (engineer) to the top (lead) where as the allocation and the payment happens in the opposite direction.

According to that research paper in Typical Software Maintenance Work flow an engineer may not find problem ticket in his best interest to report the ticket complexity truthfully and hence boost the reported value of ticket complexity for individual selfish benefits, which may lead to in-efficient ticket allocation. Hence, the central objective of the ticket allocation problem is to ensure that every individual participating in the allocation does not improve his payoff by revealing ticket complexities untruthfully.

The ticket has to be allocated to a group of engineers such that, each engineer in the proposed group makes partial contribution to solve the ticket. The engineers in the proposed group get paid for their portion of contribution to the overall problem. The decision of deciding the proposed group, their contribution and payments is called as a "Group Ticket Allocation Problem".

In this mechanism it is clear that if someone tells lies about his ticket complexity his utility decreases. When one player reveals a lie, the utilities of other players will increase.

In some papers they address this issue by proposing two Incentive Compatible mechanisms for solving group ticket allocation problem with customer delivery deadlines. They showed that these mechanisms motivate engineers individually and also in groups. So we can use these strategies to improve benefits in Software Engineering projects.

*5) Communication among stake holders*

There is an agenda planning technique with a built-in incentive mechanism, based on the VCG (Vickrey-Clarke-Groves) method from game theory, to help project managers in the engineering construction industry to create a more effective agenda [11].

Meetings are ubiquitous in projects: they foster effective teamwork and appear necessary for human connection. It is important for a team to have a forum to share concerns about project issues or progress on work that is currently underway. Some symptoms of a bad meeting are

- Low group participation
- Free riders
- A bad decision-making process
- Failure to hold a group's attention

In practice and in research, a common implicit assumption is that the meeting has a proper agenda. While all participants agree that a concise and relevant agenda leads to successful meetings, each participant prioritizes his own needs when proposing more topics to be included in the meeting agenda.

There are two main difficulties in selection of agenda items according to the paper "Improving Meeting Effectiveness by Focusing on the Agenda" [11]. The two difficulties are

- Type I Error (false negative): an important topic may be excluded from the meeting agenda,
- Type II Error (false positive): a topic with low relevance for the group may be included in the meeting agenda.

There is an interesting mechanism to overcome this as a group. Authors propose a four-step meeting agenda planning mechanism to improve meeting effectiveness, efficiency, and participation added value. The basic idea is to impose a fee on participants who bring irrelevant items to be discussed in a meeting. The process is as follows:

- Form candidate agenda
- Follow voted agenda
- Solicit additional agenda items
- Last call

Authors have successfully applied the mechanism in an actual engineering project meeting. Preliminary results show a beneficial impact on meeting effectiveness, efficiency and value added for participation. Their augmented voting agenda allows group participants to access the actual dependency between tasks which is generally noticeable only after the meeting.

In that way game theory can be used for the benefit of the software engineering processes because meetings are ubiquitous in software engineering projects.

*6) The offshore software outsourcing*

The offshore software outsourcing is client outsourcing software development to a vendor outside his country. It can be generally considered as a non-zero sum game where the gain of one player doesn't mean a loss of the other. However if both client and vendor is opportunistic it can turn in to a zero sum game. In most situations the renewal of the client-vendor relationship happens with the life cycle of the outsourced software. When this renegotiation happens Game theory concepts come in handy.

The following attributes of a game maps with this situation [4].

- Power: Power is the ability of a player to dominate the other. In the beginning of the relationship the client has the power over the vendor. But when iteration completes this power is likely to shift towards the vendor [4].
- Added value: Added value measures what each player brings to the game [4]. It is the governing factor of the power of the player (vendor or client). To reduce the added value of the vendor the client can distribute the work between many different vendors and keep this information a secret.
- Perception: Any player of a game takes actions according to the way he perceives the game. Faced the same situation two persons may make two different moves. To guess the moves of the other it is necessary to understand his perception of the game. For an example application of this factor if the client sees more value in the work done he may overpay the vendor. If the vendor sees less value he may underbid.
- Rationality: Rationality is acting in the best known way. It is a closely related concept with games. In offshore software outsourcing, client and vendors can be considered as rational partners. They direct their strategies based on how much benefit they can avail from each action they take [4].

We can use above properties of the offshore software outsourcing to model it as a game. And apply game theory strategies and concepts to handle them.

When it comes to offshore software outsourcing the communication between the two parties is very thin. Along with this fact the following explanation maps this situation to a prisoners' dilemma. When vender provides software he can provide working software (cooperate) or some code that doesn't work according to customers request (defect),the client can pay (cooperate) or not pay (defect). As the payments are done in iterations it is actually a model where prisoners' dilemma is iterated. The iterated prisoners' dilemma is a very interesting study in the Game theory. Many strategies to be used in this situation are discussed in many texts in mathematical and social aspects. They can be applied to this model.

### C. Technical aspects of Software Engineering

#### 1) Selecting a Suitable Architecture

Making a Software Architectural Design based on clients' requirements is a critical task. We can use Game Theory to obtain an optimal solution. This is known as *Quality Attribute Game* (QAG) [12] technique which enables the automation in exploring the optimum design decisions based on quality requirement values.

The main objective of the technique is that optimizing the conflicting behavior of quality attributes of an architectural design. The solving methodology is assigning relative values to the quality attributes and using problem solving concepts in Game Theory, to select the optimum architectural design. The ultimate goal is to automate the side-effects based architectural design process.

#### 2) Improving Code Quality: Identifying Extract Class Opportunities through Game Theory

Extract method refactoring is a method of improving code quality of a software program. This can be modeled as a non-cooperative game involving two players. When a particular class is given, two players compete for the methods of the original class to build two classes with high cohesion and low coupling. Adhering to the game theory building two classes is done iteratively. At every iteration a player is allowed to select at most one method for her/his class. A player selects the method to be extracted by considering the impact of adding that method in terms of cohesion and coupling of his/her class.

At the beginning the least cohesive methods are assigned to the two players. Then, the two players will iteratively select methods for their classes from remaining n-2 methods. In a given iterative a player can perform one of the following:

- Selects the method mi and yield the method mj to her opponent (i; j move);
- Selects the method mi while her opponent does not select any method (i; null move);
- Does not select any method while her opponent selects the method mj (null; j move)

[2] pp2.

What action to be made is decided by finding the Nash equilibrium in the payoff matrix.

Once the action is identified the player selects the appropriate method (if any) and builds his/her class. The game is over when all the classes in the original class are assigned to one of the two classes. (How to compute the payoff matrix is mentioned in [2])

## III. Issues Related to the Usage of Game Theory

Game Theory is a mathematical tool which helps to make rational decisions. The first known discussion of Game Theory was raised in 1713 by James Waldegrave [13]. Since then, the subject Game Theory has evolving by more focusing on economics, computer science, biology, philosophy and political science. But, here we are focusing on Software Engineering processes which are not mature enough and evolved frequently. For an example, initially waterfall method used heavily, today software engineers prefer iterative methodologies to address rapid changing of business requirements. Therefore adapting new subject, game theory into Software Engineering is not still popular. But there is a huge potential to use game theory in Software Engineering projects in the future.

## IV. Conclusion and Future Work

By reviewing the available literature, it can be observed that the requirement of finding applications of game theory in software engineering has been studied to a certain extent, but yet has a far way to go. It is interesting to see that the game theories' penetration in to almost every aspect of the software engineering, even though many of those are in very immature levels in real world applicability. Therefore considering the future of this topic, it is suggestive to having a big picture of software engineering can emerge enormous potential of developing collaborative and productivity related applications in the software industry.

## V. References


[1] T. L. Turocy and B. von Stengel 2001. "Game Theory". CDAM Research Report LSE-CDAM-2001-09

[2] G. Bavota, R. Oliveto, A. de Lucia, G. Antoniol, and Y. Gueheneuc. 2010. "Playing with refactoring: identifying extract class opportunities through game theory". In proceedings of the 2010 IEEE International Conference on Software Maintenance (ICSM '10). IEEE ComputerSociety,Washington,DC,USA,1-5. DOI=10.1109/ICSM.2010.5609739

[3] O.Hazzan and Y Dubinsky. 2004. "Social perspective of software development methods: the case of the prisoner dilemma and extreme programming"

[4] N. V. Oza. 2006. "Game theory perspectives on client – vendor relationships in offshore software outsourcing", EDSER '06 Proceedings of the 2006 international workshop on Economics driven software engineering research ACM New York, NY, USA.

[5] J. Buisman and C. Wohlin, 2003 "Using game theory to study bidding for software projects", in proceedings EASE: Empirical Assessment and Evaluation in Software Engineering, Keele, UK, 2003.

[6] J. B. Berk, E. Hughson, K. Vandezande, "The price is right, but are the bids? an investigation of rational decision theory", The American Economic Review, Vol. 86 No. 4, September 1996, pp 954-970

[7] M. Griffiths. 2010." Aligning PMOs using game theory" [Online]Available:http://leadinganswers.typepad.com/leading_answers/2010/11/aligning-pmos-using-game-theory-.html [accessed:20th January 2012]

[8] Standish Group Chaos Report (2004) [Online] Available: http://blog.standishgroup.com/ [accessed:20th January 2012]

[9] M. Grechanik and D. E Perry. "Analyzing software development as a noncooperative game". The Center for Advanced Research In Software Engineering (ARISE) The University of Texas at Austin.

[10] K. Subbian, R. Kannan, R. K Gautam and Y Narahar 2008."Incentive compatible mechanisms for group ticket allocation in software maintenance services" 14th Asia-Pacific Software Engineering Conference.

[11] A. C. B. Garcia, J. Kunz, and M. Fischer. 2004. "Cutting to the chase: improving meeting effectiveness by focusing on the agenda". In proceedings of the 2004 ACM conference on Computer supported cooperative work (CSCW '04). ACM, New York, NY, USA, pp346-349.

[12] E. H. Hagen and P. Hammerstein 2005. "Game theory and human evolution: A critique of some recent interpretations of experimental games" .Theoretical Population Biology 69 (2006) pp.339–348

[13] K. K. Vajja, and T. V. Prabhakar. 2009. "Quality attributes game: a game theory based technique for software architecture design". In Proceedings of the 2nd India software engineering conference (ISEC '09). ACM, New York, NY, USA, pp.133-134.