Improving Software Development Process through Economic Mechanism Design

Murat Yilmaz[1] and Rory V. O'Connor[2,3] and John Collins[4]

[1] Lero Graduate School in Software Engineering, Dublin City University, Ireland
Murat.Yilmaz@computing.dcu.ie
[2] Dublin City University, Ireland
[3] Lero, the Irish Software Engineering Research Centre
roconnor@computing.dcu.ie
[4] University of Minnesota, Minneapolis
jcollins@cs.umn.edu

Abstract. We introduce the novel concept of applying economic mechanism design to software development process, and aim to find ways to adjust the incentives and disincentives of the software organization to align them with the motivations of the participants in order to maximize the delivered value of a software project. We envision a set of principles to design processes that allow people to be self motivated but constantly working toward project goals. The resulting economic mechanism will rely on game theoretic principles (i.e. Stackelberg games) for leveraging the incentives, goals and motivation of the participants in the service of project and organizational goals.

1 Introduction

In this paper, we describe a novel approach by using economic mechanism design in software engineering to find ways to assemble the rules of interactions and influence the behavior of rational players, and further, by game theoretic principles to determine how a decision among a group of participants can be organized to maximize the social and organizational productivity, and therefore welfare of a software project.

1.1 Software Process

The software development process explains the methods and procedures which an organization and individuals have to follow to create software products and services [1]. A software process is a dynamic vehicle formed from a group of interrelated activities employed by a project or an organization [2]. These activities not only produce products or services but also provide a road map for the software development within the expected schedule and budget [3]. Each process can be decomposed into its activities and each activity is characterized by its tasks (smallest unit of work) [2]. In the generic model below (see figure 1), tasks are the atomic structures designed to process the resources into products.
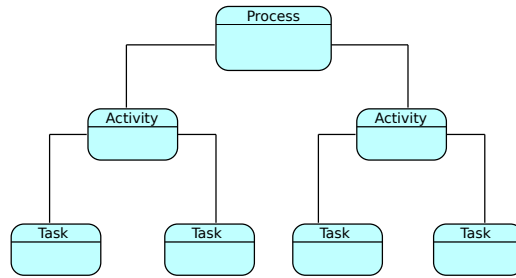
Fig. 1. The structure of a generic process from [4].

Because software organizations can be considered as social systems built on people's skills, these tasks and activities are tightly coupled with participants that interact according to "certain theories of actions" [5]. Based on their resources, values, goals, and skills, software organizations directly use or sometimes tailor a process regarding to their needs [6]. Therefore, all software organizations use a kind of process.

1.2 Software Process Models

A software process model is an *abstract representation* of a process and its components which imposes the tasks and activities of software development [7]. These models help practitioners to manage various aspects through the development cycle. Over the years, many different process models have been introduced including; waterfall model [8], iterative enhancement model [9], and spiral model [10], and ISO/IEC 12207 (an international standard for software life-cycle processes). ISO/IEC 12207 [2] furnishes a comprehensive set of process (i.e. activities and task structures) endowed with Plan-Do-Check-Act (PDCA) cycles for administering the engineering activities of a software system.

Although there are many different approaches for modelling a software process (e.g. work-flow diagrams, Petri nets, finite state machines), to the best of our knowledge, there is neither a process model nor a paradigm which is based on the social modelling of the human behaviour.

In this section, we briefly discuss the software process and process models, and the rest of the paper is structured as follows. Section 2 provides an overview on software process economics, some background about game theory, application of game theory and economic decision making in software engineering research,and followed by an introduction about mechanism design theory. Section 3 contains our approach for software development process as a mechanism design problem and a sample mechanism. Section 4 includes a conclusion and ideas about future work.

2 Software Process Engineering Economics

Software Process Engineering deals with the global insights of software processes by creating, designing and optimizing (a set of) solutions to obtain performance targets (e.g. cost effective and sustainable software development) of a software organization [11].

Levy [12] argues that although software economics is valuable in estimating project costs by applying economic methods to software projects, researchers should also create models for economic analysis of software engineering projects. As a matter of fact, one different view of software economics is that participants structure the actual products and the processes by their (investment) decisions based on the options and available resources [13]. Although, it is not a common tool in software engineering research, as in economics game theory has been used to a great extent for understanding varieties of entities and their behaviours. Besides, it is frequently used for mathematical modelling of human choices, and social interactions. Most of the software development processes are based on team work and collaboration so it is considered as the output of a set of social activities [14]. In other words, software development teams consist of interacting actors or entities with social concerns (e.g. motivation or knowledge sharing) where economic and social modelling [15] should be useful to create a game theoretic model based on a life-cycle of a development process.

To sum up, an economical view of software engineering takes value creation activities and value based project planing into account to support software process evolution by value based process optimization. The two identified goals of software process engineering suitable for a game theoretic model includes; (i) allocating the organizational resources regarding to the task and activities of a software project, (ii) maintaining a stable social environment by addressing technical or managerial conflicts to achieve and keep higher productivity. A game theoretic model will be helpful to identify some potential points of concern (i.e. conflicts, social dependencies, and actions) among the software organization and to accomplish its objectives by creating several types of mechanism (i.e. regulatory rules).

2.1 Game Theory

Game theory is a well developed theory for describing the interactions of rational, independent agents in a variety of settings used for creating approaches in fields including, economics, computer science, social science, political science, and biology [16].

Game theory (i.e. interactive decision theory) investigates the outcomes of interaction of entities. It is a collection of analytical methods or tools based on mathematical models to define or observe social situations (e.g. conflicts) [16]. To this end, game theory outlines interaction of people in terms of mathematical game forms. These forms consist of players (participants or actors), rules required for their interactions, actions or strategies (strategy profiles) of participants, and have definition of outcomes (i.e. payoffs) of their actions. One of the basic

assumption in classical Game Theory is that participants are rational, i.e. follow the rules and play to win [17]. The types of common game forms, including; (1) Non-cooperative games where participants only acts according to their benefits, (2) Cooperative games where participants inclined for cooperative behaviour (i.e. cooperation is used as the main motivator), (3) Zero Sum games where one of the participants should win the game while other(s) lose, (4) Constant Sum games where the reward for each participant is constant [18].

2.2 Game Theory in Software Engineering Literature

Because it is important to explore all the alternative paths (e.g. creating a prototype versus the whole product) as measurable returns, it is becoming difficult to ignore the fact that the economic value of software decision making should be examined. As has happened in many other fields like sociology, economics or computer science, there are some approaches for economic decision making in software engineering research.

Theory W [19] is a software project management theory that has its roots in economics and decision analysis. It not only helps managers (social planners) to identify stakeholders and their goals but also suggests ways to resolve any conflicts among them by proposed methods (e.g. risk management). Based on participants preferences, Lagesse [20] creates a game-theoretical model for assigning tasks in software projects. Grechanik and Perry [21] argues that software development is a non-cooperative game in which there may have some observable goal conflicts between individuals and software organizations. Cockburn [22] claims that software development is a form of a game, which is based on communication and coordination skills where project resources are one common constraint. Baskerville et al. [23] investigates high speed internet development which is based on rapid consumption of resources. Therefore, it depends more on the developers and their decision making capabilities.

By using the theory of real options, Sullivan *et al.* [24] present an economical approach for valuation of software design decisions which are somewhat similar to capital investment decisions. Their aim is to optimize strategies by measuring strategic outcomes. Vajja and Prabhakar [25] consider a design decision analysis for architecture design problems by understanding conflicting situations created by quality attributes, and seeking solutions using game theory. A recent study by Sazawal and Sudan [26] involves with game theory to model decision making in software design to shape its evolution. They design a model called software design evaluation game which is playable by a developer and a customer. Moreover, they propose that software development teams should use a lightweight game theoretical analysis technique for situational analysis.

Social dilemmas are situations which arisen from the fact that there is a conflict between collective and private interests (mostly long term versus short term). In other words, participants may discover a better or a more feasible alternative action to gain more by following their self interests other than team based contributions. Prisoners Dilemma is a simple framework which has been

used frequently by researchers to observe conflicting situations. Hazzan and Dubinsky [27] draws our attention to situations where Prisoners Dilemma can occur in software development. In particular, they analyze cooperation in extreme programming practices (e.g. pair programming). Feijs [28] provides a game theoretic model for the tester and the programmer where he finds that the definitions of outcomes could cause a hidden game of Prisoners Dilemma. In addition, Oza [29] investigates client vendor relationships in offshore outsourcing by using Prisoners Dilemma.

From software organizations perspective, these findings suggest that game theory is applicable to various software engineering problems. As the outcome of a software development process depends on many structural assessments on a a social landscape, the value of understanding how individuals can be promoted for cooperation is vitally important.

2.3   Mechanism Design

A mechanism can be defined as an organization, a procedure or a communication system which takes the required information from participants as inputs and determines a social outcome. The goal of mechanism design is to establish the rules of interaction (e.g. protocols, regulatory rules, etc.) to satisfy the desired conditions (e.g. control and coordinate the flow of the economic activity) [30]. These rules can be used for revealing true preferences of self interested actors, optimizing usage of resources, exchanging information or coordinating economic actors [31].

Hayek [32] developed the idea to view social organizations as mechanisms for communication and information exchange. Hurwicz [30] introduces the concept of economic mechanisms to model organizations where participants communicate, and exchange information with each other. Further, he coined the term *incentive compatibility* which ensures that self-interested individuals can be motivated to reveal their true preferences and their private information. His further research aims to model organizations based on communications and actions that are available to each participant in an institution. Harsanyi [33] developed a model based on the theory of Bayesian games, (i.e., games with incomplete information). He investigates situations where individuals have different information. In particular the issues where participants have uncertainty about other participants' information (while the the rules of the game form is known by all the participants). Moreover, he worked on models of incomplete information based on issues of modeling participants' actions in terms of each others' in social organizations.

3   Game Theoretic Perspective

We assume software organizations act as social (co-evolving) ecosystems. An ecosystem can be any kind of environment with different type of entities which are constantly interacting. Hence the relations among them are identified by their

interdependencies. Mitleton-Kelly [34] defines organizations as social ecosystems where connectivity and relationships affects the actions, behaviours and decisions of its members.

From a social perspective, a software process can be considered as a work flow in social information streams. Consequently, a software organization can be defined as combinatorial networks of people connected with various information. Therefore, in order to address the adversities associated with human actors and their connections, it is important to investigate the interactions of these actors in these information streams. It has been suggested that improving the quality of social interactions among the participants is an important factor for accomplishment of organizational goals [35]. The social information network created by the software organization should be dominated by actions and characteristics of actors (e.g. individualism, rationality) where the conflicts between individual and team goals may hinder software development process [21].

A task is an atomic unit executed by individuals and teams of actors that are interacting and making several investment decisions. Further, these decisions are made regarding to the characteristics of these actors e.g. background, skills, values and motivational needs. Beecham *et al.* [36] identifies several social factors effecting software engineer characteristics. For example, software practitioners prefer challenging tasks and therefore recognition in their work, stability in their organization, and most importantly they prefer to be socially identified with their team or group [36].

Software development is inherently a complex process in which a common element of uncertainty is primarily caused by humans [37]. Software development can be considered as the interaction among several actors that are playing separated roles with different behaviour patterns. The roles typically include developers (designers and programmers), system analysts, testers, and customers or users. Predictably, interpersonal conflicts between these roles are unavoidable during the process [38, 39]. Therefore, the participants of software development processes need to be identified by their private information, that is their *goals, aims, preferences and skills* (We will refer to those as "types" in this paper).

Here we describe a generic view for the software process decisions with four major components (see figure 2); (i) outcomes are the expected outputs achieved during the execution of a decision; (ii) states are an observable set or sequence of attributes (e.g. quality, stability, etc.) of a decision identified by a measurement or an observer at a certain time, (iii) rules are the constraints that are predefined during the creation of a decision, (iv) actions are the activities chosen with the evaluation of the circumstances in various situation for an optimal outcome. Moreover, we define software development tasks in terms of participants' collective decisions, i.e. their strategies, actions, interaction types (i.e. cooperation or competition) and private information (i.e. types). We assume strategies could be formed from one or more strategies and they are decomposable. Strategies are combined to form an action or an action set. The term "action" is used to represent grounded states or collection of states and their interactions. In other words, interactions are formed from the combination of actions, individual types

and ongoing situations. We presume observable events are emerged from these set of intersections. A set of outcomes arise from different situations, sometimes as a cooperation and sometimes in form of a competition.
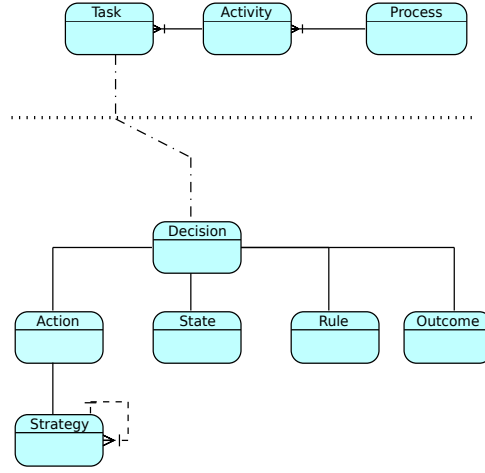

Fig. 2. Extended generic process for software development

3.1 Software Development Process as an Economic Mechanism

Information has an economic value, which should enable software decision makers to make better decisions for maximizing the productivity of software development. Therefore, we consider software development process as an economic activity (i.e. production/distribution of project tasks and organizational services) based on an information exchange economy. From this perspective, it can be understood as a social ecosystem that can be materialized as interconnected networks of actors, their actions and relationships. The goal is to create incentives to support and incrementally feed the participants interests.

Our setting relies on; (i) multiple participants with different individual types that are continually adapting to the environment and each other, (ii) participants have asymmetric imperfect information, they know their own type, but not all the other types (i.e. goals, aims, preferences, skills), in other words, participants have insufficient information about the characteristics of other players, (iii) therefore, communication is constrained.

Software development process can be considered as an economic activity inside an information exchange economy

$$\xi_{software} := \{t_i, \theta_i, u_i, s_i\}_{i \in I} \otimes \mathcal{A} \tag{1}$$

which is an economy with a finite set of actors i with a set of possible actions $\mathcal{A}$, for each participant, T is a set of tasks of project, where t represents tasks assigned for each participant, an actor chooses a strategy s_i from a finite set of strategies S, where θ_i is participants i's type space with a possible probability distribution p_i over types, utility function u_i shows participant's payoffs.

We define productivity of a participant, x;

$$x = \phi(t_i, \theta i) \tag{2}$$

$\phi : T \otimes \Theta \rightarrow \Re$, where Θ is a set of type profile space of the participants' types $\theta = (\theta_1,....,\theta_n)$, hence we can calculate the total productivity as;

$$\sum_{i \in I} \phi(t_i, \theta i) \tag{3}$$

In the light of these remarks, we assume organization of software development can be viewed as an economic mechanism where manager (i.e. social planner) allocates resources and adjusts the incentives and disincentives to maximize the likelihood of a desirable outcome.

A perfect information exchange in this information economy will be a mapping from types to actions, $s_i : \theta_i \rightarrow S_i$, however, capturing the true preferences of participants is a challenging process.

A conceptual solution for social planing problem in a software development organization can be solved by;

- First, requesting the participants to reveal their types, θ_i
- Second, using them directly to compute the intended social outcome.

On the other hand, the 'participants' preferences over the set of actions $\mathcal{A}$ depend on the realization of types Θ. It won't yield an utility of intended social outcome if it results from participants taking actions that maximize their own expected utilities. Actor i may reveal untruthful type to manager (i.e. social planner), however, the manager can tackle this problem by forming a Stackelberg game.

A Sample Mechanism

During software development process, individuals' incentives may be inconsistent with expected efficiency, so here we form a sample mechanism appropriate for organization of software development. We form a Stackelberg game [40], which has been well-studied in the game theory literature [41, 42].

Stackelberg game (interaction model) is based on leaders (a dominant player type) who simultaneously choose a strategy first, and then regarding the strategy chosen by their leaders, the followers act to maximize their payoffs [40]. Similarly, we will use a mechanism, in which a leader (usually a project leader is responsible for assigning tasks to followers), aims to maximize her profit subject to all of other team members (followers). The problem is then to establish a

strategy for leaders that motivates these followers to react in a way such that an equilibrium relative to the leader's strategy not only optimizes their rewards but also increases the overall project outcomes [43].

Software teams consist of team leaders and team members (followers). In our game (form), it is assumed that there is only one type of leader (team leaders), although there are multiple followers (a category that includes programmers, systems analysts, designers, testers).
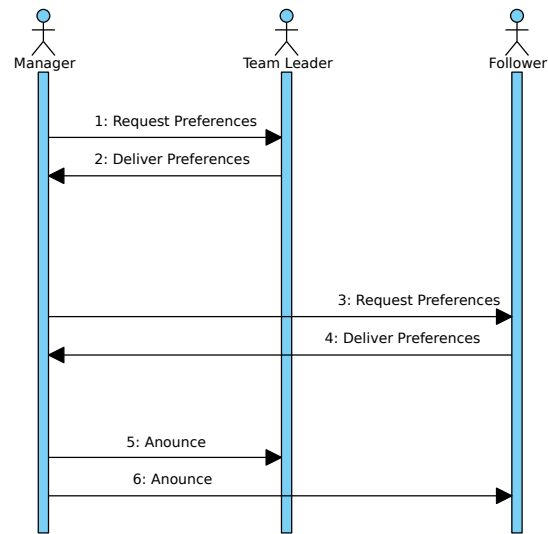

Fig. 3. Information Exchange in Stackelberg form in a Software Organization

A sample mechanism proceeds through a sequence of states by considering the fact that motivating team members is to ask them what or who motivates or demotivates them. Our aim is to classify participants as leaders and followers and decouple them to understand their real preferences easily (see figure 3).

- Team leaders identify available motivators of followers (e.g. rewards, hidden goals) and demotivators (e.g. social interaction problems, problems among the people) and align them with their team development activities that will help for the success of the components they will deliver.
- From their perspectives, team leaders create some priorities among their members that incentivize their team's development effort (e.g. a reward for teams after an iteration) for the software development process.
- For each activity, they classify these priorities to highlight the most valuable ones for the individuals and team.
- From their perspectives, team members also identify and classify the most valuable motivators and demotivators. Later, all information is submitted to the manager.

- The manager, as a social planner, invites team leaders to reveal their preferences (their privately held information about motivators) simultaneously.
- The manager broadcasts a part of this information (i.e. list of prioritized motivators) among the followers.
- Followers respond to this by revealing their preferences about these motivators simultaneously. This process continues until all the participants announce their preferences.
- After receiving all the preferences from individuals, manager form and announce an organizational decision.

4 Conclusions and Future Work

Software development requires teams of self-interested individual actors to contribute effectively to organizational goals. Organizations that fail to account for the motivations of individual participants often experience difficulty accomplishing their goals [44]. Game theory suggests that we view the organization and its goals from the standpoint of individual rational actors, who choose actions that maximize their expected utilities, subject to their incomplete knowledge of the motivations and likely actions of others, and their limited ability to predict future outcomes. The challenge for software organizations is to allocate resources and create incentives and disincentives in such a way that participants are motivated to take actions that contribute effectively to organizational goals.

The aim of this research is to model software development as economic activity, and the organization of software development as an economic mechanism problem. We have described how software development might be viewed as a Stackelberg game, in which the individual motivations of the participants are discovered and used to tune the organization and its relationship to the participants in a way that can maximize the expected outcome of the project as a whole.

In order to evaluate the value of this approach, it will be important to conduct rigorous analysis of actual software projects, understanding their underlying economic mechanisms and discovering how the participants view and interact with those mechanisms. Next, we will apply our revised models to one or more actual projects and analyze their effectiveness. This may require separate "control" and "test" projects, because the effort to clearly understand a project's goals, without taking any other action, may itself have significant impact on a project.

Acknowledgments

References

1. O'Connor, R.V.: Human aspects of information technology development. International Journal of Technology, Policy and Management **Vol. 8, No. 1** (2008)

2. ISO/IEC: Amendment to ISO/IEC 12207-2008 - Systems and software engineering Software life cycle processes. (2008)
3. Pressman, R.S., Ince, D.: Software engineering. McGraw-Hill (2000)
4. Singh, R.: International Standard ISO/IEC 12207 software life cycle processes. Software Process Improvement and Practice **2** (1996) 35–50
5. Conradi, R.: Software process improvement: results and experience from the field. Springer-Verlag New York Inc (2006)
6. Persse, J.R.: Process Improvement Essentials. O'Reilly Media, Inc. (2006)
7. Acuna, S.T., Juristo, N., Moreno, A.M., Mon, A.: A Software Process Model Handbook for Incorporating People's Capabilities. Springer-Verlag New York, Inc. (2005)
8. Royce, W.: Managing the development of large software systems. In: Proceedings of IEEE Wescon. Volume 26. (1970)
9. Basili, V.R., Turner, A.J.: Iterative enhancement: A practical technique for software development. IEEE Transactions on Software Engineering **4** (1975) 390–396
10. Boehm, B.: A spiral model of software development and enhancement. Computer **21** (1988) 61–72
11. Madachy, R.J.: Software Process Dynamics. Wiley-IEEE Press (2008)
12. Levy, L.S.: Taming the tiger: software engineering and software economics. Springer-Verlag New York, Inc. (1987)
13. Biffl, S., Boehm, B., Erdogmus, H.: Value-based software engineering. Springer-Verlag New York Inc (2006)
14. Dittrich, Y., Floyd, C., Klischewski, R.: Social thinking-software practice. The MIT Press (2002)
15. Yu, E.: Social Modeling and i*. Springer (2009)
16. Dixit, A.K., Skeath, S.: Games of Strategy. W. W. Norton & Company (1999)
17. Osborne, M.J., Rubinstein, A.: A course in game theory. MIT Press (1994)
18. Binmore, K.G.: Playing for real. Oxford University Press US (2007)
19. Boehm, B., Ross, R.: Theory-W software project management principles and examples. Software Engineering, IEEE Transactions on **15** (1989) 902–916
20. Lagesse, B.: A Game-Theoretical model for task assignment in project management. In: 2006 IEEE International Conference on Management of Innovation and Technology, Singapore (2006) 678–680
21. Grechanik, M., Perry, D.E.: Analyzing software development as a noncooperative game. In: IEE Seminar Digests. Volume 29. (2004)
22. Cockburn, A.: Agile software development: the cooperative game. Addison-Wesley (2007)
23. Baskerville, R.L., Levine, L., Ramesh, B., Pries-Heje, J.: The high speed balancing game: How software companies cope with internet speed. Scandinavian Journal of Information Systems **16** (2004) 11–54
24. Sullivan, K., Chalasani, P.: Software design decisions as real options. (1997)
25. Vajja, K.K., TV, P.: Quality attribute game: a game theory based techniquefor software architecture design. In: Proceeding of the 2nd annual conference on India software engineering conference, Pune, India, ACM (2009) 133–134
26. Sazawal, V., Sudan, N.: Modeling Software Evolution with Game Theory. (Trustworthy Software Development Processes) 354–365
27. Hazzan, O., Dubinsky, Y.: Social perspective of software development methods: The case of the prisoner dilemma and extreme programming. In: Extreme Programming and Agile Processes in Software Engineering. Springer (2005) 74–81
28. Feijs, L.: Prisoner dilemma in software testing. Computer Science Reports **1** (2001) 65–80

29. Oza, N.: Game theory perspectives on client: vendor relationships in offshore software outsourcing. (2006) 54
30. Hurwicz, L., Reiter, S.: Designing economic mechanisms. Cambridge Univ. Pr. (2006)
31. Myerson, R.B.: Perspectives on mechanism design in economic theory. American Economic Review **98** (2008) 586–603
32. Hayek, F.A.V.: The use of knowledge in society. American Economic Review **35** (1945) 519530
33. Harsanyi, J.C.: Games with incomplete information played by "Bayesian" players, I-III: part i. the basic model&. MANAGEMENT SCIENCE **50** (2004) 1804–1817
34. Mitleton-Kelly, E.: Complex systems and evolutionary perspectives on organisations. Emerald Group Publishing (2003)
35. Ryan, S., O'Connor, R.V.: Development of a team measure for tacit knowledge in software development teams. Journal of Systems and Software **82** (2009) 229–240
36. Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H.: Motivation in software engineering: A systematic literature review. Information and Software Technology **50** (2008) 860–878
37. Ziv, H., Richardson, D., Klosch, R.: The uncertainty principle in software engineering. In: Proceedings of the 19th International Conference on Software Engineering (ICSE'97). (1997)
38. Barki, H., Hartwick, J.: Interpersonal conflict and its management in information system development. Mis Quarterly (2001) 195228
39. Zhang, X., Dhaliwal, J.S., Gillenson, M.L., Moeller, G.: Sources of conflict between developers and testers in software development. AMCIS 2008 Proceedings (2008) 313
40. Stackelberg, H.V., Peacock, A.T.: The theory of the market economy. Oxford University Press (1952)
41. Bloem, M., Alpcan, T., Basar, T.: A stackelberg game for power control and channel allocation in cognitive radio networks. In: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools, Nantes, France (2007) 1–9
42. Pita, J., Jain, M., Ordez, F., Tambe, M., Kraus, S., Magori-Cohen, R.: Effective solutions for real-world stackelberg games: when agents must deal with human uncertainties. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, Budapest, Hungary (2009) 369–376
43. Ichiishi, T., Yamazaki, A.: Cooperative Extensions of the Bayesian Game. World Scientific Publishing Company (2006)
44. Rasch, R.H., Tosi, H.L.: Factors affecting software developers' performance: An integrated approach. MIS Quarterly **16** (1992) 395–413