Multi-user Variability Configuration: A Game Theoretic Approach

Jesús García-Galán
Dept. Languages
and Computer Systems
University of Seville
Seville, Spain
Email: jegalan@us.es

Pablo Trinidad
Dept. Languages
and Computer Systems
University of Seville
Seville, Spain
Email: ptrinidad@us.es

Antonio Ruiz-Cortés
Dept. Languages
and Computer Systems
University of Seville
Seville, Spain
Email: aruiz@us.es

Abstract

Multi-user configuration is a neglected problem in variability-intensive systems area. The appearance of conflicts among user configurations is a main concern. Current approaches focus on avoiding such conflicts, applying the mutual exclusion principle. However, this perspective has a negative impact on users satisfaction, who cannot make any decision fairly. In this work, we propose an interpretation of multi-user configuration as a game theoretic problem. Game theory is a well-known discipline which analyzes conflicts and cooperation among intelligent rational decision-makers. We present a taxonomy of multi-user configuration approaches, and how they can be interpreted as different problems of game theory. We focus on cooperative game theory to propose and automate a tradeoff-based bargaining approach, as a way to solve the conflicts and maximize user satisfaction at the same time.

I. Introduction

Variability is a property of software systems that provides them with the ability of changing and being customized according to the specific user needs. Highly-configurable systems like Debian OS or Android, adaptive systems like smart homes, cloud services like Amazon Web Services or *Software Product Lines (SPLs)* are examples of *variability-intensive systems*. Variability configuration is the process where one or more users make decisions about system variability. It is a complex process since *(1)* user decisions may be invalid and *(2)* decisions can lead to conflicts when multiple users take part. That leads variability configuration to embrace automated support.

Several approaches have been made along the last years to assist the configuration process. The main issue they face is dealing with multi-user conflicts. Most of the proposals avoid them, defining mutual-exclusion criteria. Staged configuration approach [3] proposes an step by step configuration process, where each user refines previous user decisions. Collaborative configuration [7] improves staged configurations by splitting the decision space in disjoint areas, so several users can configure at the same time without conflicts. These approaches avoid conflicts, but they also limit the space in which users make decisions. Consequently, they limit user satisfaction, since users

cannot make decisions they would like to make. Recently, conflict resolution approaches have been also proposed [16]. However, they do not either consider user satisfaction to solve conflicts.

In this paper, we interpret multi-user configuration of variability-intensive systems as a game theoretic problem. Game theory is a discipline which deals with "*the study of mathematical models of conflict and cooperation between intelligent rational decision-makers* [8]. Depending on the game nature and the users' aiming, games are classified into two different groups: cooperative or non-cooperative games. In non-cooperative games, users look for maximizing their satisfaction in an individual way, while in cooperative games users can negotiate among them to obtain a better satisfaction value. Non-cooperative games have been traditionally more addressed than cooperative ones because of their similarities with economic environments. However, a cooperative approach provides more satisfactory solutions in many scenarios, like the classic prisoner's dilemma [6].

We focus on multi-user configurations of *Feature Models (FMs)*, a widespread type of variability model. Our goal is solving multi-user configuration conflicts maximizing user satisfaction. This work presents two main contributions. Firstly, we identify multi-user configuration conflicts as a non solved problem, and we describe it in terms of a game theoretic problem. In this sense, we also present a taxonomy for multi-user configuration based on different 'dimensions', and which has its analogous correspondence in game theory. And secondly, we propose an automated bargaining process, inspired in cooperative game theory, to achieve conflict-free and satisfactory configurations. We present a tradeoff system to let users express the decisions they would give up, how it affects their satisfaction and a possible compensation. In terms of automated analysis, we propose a new analysis operation to solve conflicts in multi-user configurations considering user satisfaction. We develop a prototype of such analysis operation to ensure that our approach is feasible.

The rest of the paper is structured as follows. Section II presents variability configuration background. Multi-user configuration problem is described in Section III, together with a case study. Our proposal is discussed in Section IV, where we also revisit the case study. Related work is exposed in Section V, and finally conclusions and further work are summarized in Section VI.

II. Background

Variability-intensive systems are described by means of variability models. *Feature Models (FMs)* [4] are the most widespread models, due to their multiple extensions and generalized usage. Figure 1 shows an example of a smart home as a FM. A basic FM is composed by a set of hierarchically arranged features, which are represented by nodes. These features can be connected to other features by means of relationships. Any FM has a root feature that represents the whole functionality of the system. The root feature is refined in child features, which decompose the functionality of the root feature into subfeatures. This refinement process is repeated for the child features to conform a tree-like structure. Hierarchical relationships are individual (*mandatory* and *optional*) or grouped (*alternative* and *or*). Although the hierarchical structure helps to represent the feature refinement, it can hinder the representation of constraints that affect features in different branches of the tree. In these circumstances, cross-tree constraints (*depends* and *excludes*) can be used.

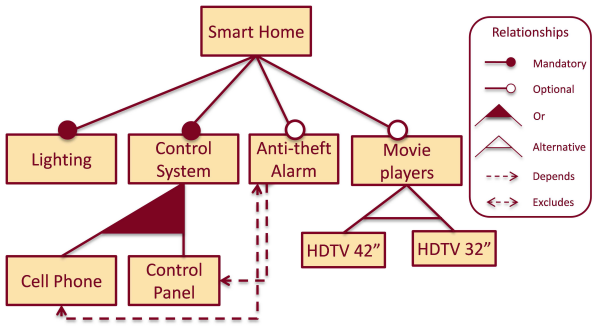

Fig. 1. Example of a smart home FM

FMs are useful for the variability *configuration process*. This process consists on one or more users making decisions about elements of the system. In a FM, these elements are essentially features, what drives to a so-called feature-based configuration process. That is, a user decides if a feature is selected, removed, or if she does not care about it. The result of this variability configuration process is a *configuration*, which is composed by pairs of elements and decisions. In this paper, we use a two set notation to represent feature-based configurations, a set S for selected features, and a set R for removed features. If a configuration assigns a value to every feature in the FM, we say it is a full-configuration. For instance, for the FM in Figure 1, S_1={`Smart Home, Lighting,Anti-theft Alarm`} R_1={`HDTV32`} is a configuration, but not a full-configuration. Depending on the FM relationships and user decisions, a configuration is valid or invalid. A valid configuration is a configuration which does not violate any relationship within the FM. For example, while the former configuration is valid, a configuration determined by S_2={`Smart Home, Anti-theft Alarm`} R_2={`Lighting`} is invalid because Lighting is a mandatory feature and its state is `removed`. When different users take part in a configuration process, it is known as a *multi-user configuration process*. This makes sense for large systems where each user is a specialist of a specific area, or for systems or devices shared by several users. In this kind of configuration, the decisions of different users may lead to conflicts, e.g. if `User A` selects `Anti-theft Alarm` and `User B` removes it. Two main approaches have been proposed to avoid conflicts in a multi-user configuration process (Figure 2). Czarnecki et al. [3] proposes step by step configurations, in such way that just a single user can configure the FM at the same time. Each user refines the decisions of the previous users until they obtain the desired result or a full-configuration. Mendonça et al. [7] proposes an improvement of this method, collaborative configurations, which let several users configure concurrently disjoint areas of the FM. However, a general approach where different users configure concurrently overlapped areas still lacks.

III. The problem

The problem we face up in this work is the general approach of multi-user configuration process, shown in the right side of Figure 2: any user can configure any area of the system and in any order. The classic inputs of the multi-user configuration process are the variability-intensive system and a set of user decisions, producing

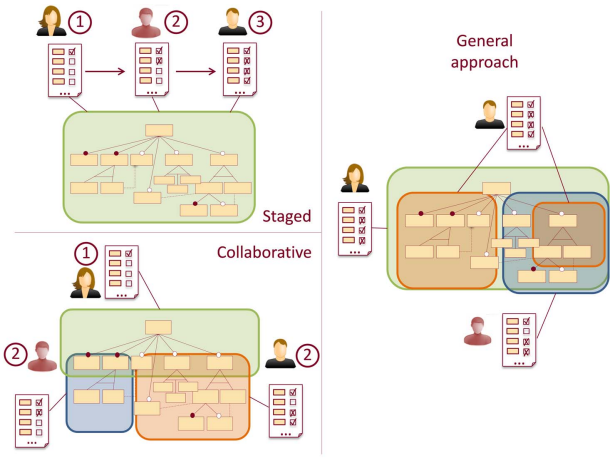

Fig. 2. Staged, collaborative and general multi-user configuration approaches

a valid state of the system as output. We model variability-intensive systems as FMs, and user decisions as FM configurations. The output should be a valid configuration, i.e., a configuration which satisfies all FM constraints, and which also satisfies all users needs in the best way.

The main issue that can arise when several users make decisions in the same configuration process is the appearance of conflicts. Such conflicts can be categorized in two groups: contradictions and constraint violations. A contradiction is conformed by a set of decisions which remove all possible values from an element domain. For example, if `User A` selects `Anti-Theft Alarm`, and `User B` removes the same feature, there is a contradiction between these two decisions. A constraint violation is given when one or more decisions do not satisfy all relationships and constraints. For example, if `User A` selects `Anti-Theft Alarm`, and `User B` selects `Cell Phone`, a constraint violation arises since there is an exclusion constraint between both features (Figure 1). This kind of validation and the process to solve these conflicts manually is tedious and error-prone for small sized FMs and few users, becoming an infeasible task for real-world cases.

We consider that multi-user configuration process has several dimensions which lead to similar but also different processes. Following, we identify such dimensions and present a brief taxonomy:

- *Mutual exclusion* vs *concurrent configuration*: a mutual exclusion approach does not allow different users to configure common areas concurrently. By contrast, concurrent configuration let users to configure without this limitation.
- *Transactional* vs *interactive*: a transactional approach groups all the decisions of each user in a transaction, in an atomic way. In an interactive approach each user decision is considered apart, while users watch the decisions of the rest of people.
- *Customized view* vs *whole view*: depending on user interests or knowledge, a customized view can be provided to ease configuration tasks, instead of a whole view of the system.
- *Decisions* vs *preferences*: while decisions express hard constraints which must be satisfied in every case,

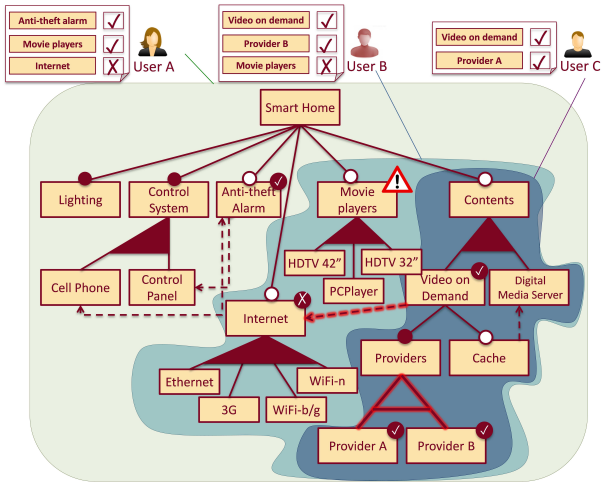

Fig. 3. Three users configuration scenario of a smart home

preferences express soft constraints which increment user satisfaction if satisfied.

The described proposals in Section II, staged and collaborative configuration, avoid conflicts but present two significant drawbacks. Firstly, both are mutual exclusion approaches, which implies that users satisfaction is affected. In the case of staged configurations, just the first user of the configuration process is totally free to make any decision. The following users are limited to decide about the set of elements that are still unaffected by previous decisions. For collaborative product configuration, users can only configure limited areas of the system. Both staged and collaborative configurations, in different ways, restrict users decision-making space. And secondly, these approaches are not applicable in scenarios which require a concurrent configuration process. SPLs configuration processes usually are performed offline. Hence, these approaches make sense for them. However, systems like smart homes are configured at runtime and in a concurrent way. For instance, several users may be interested in configuring shared areas, like connectivity or lighting.

A. *Case study*

We present a smart home case study in order to clarify the problem. In this scenario, depicted in Figure 3, three different users configure the system: `User A`, `User B` and `User C`. Each one is interested in a specific area of the FM. `User A` configures the whole model, `User B` is interested in `Internet`, `MoviePlayers` and `Contents` their subfeatures, and `User C` just in `Contents` and their subfeatures.

We can detect easily a contradiction between `User A` and `User B` about `Movie players` feature. Moreover, there are two more conflicts caused by FM relationships. `Video on demand`, selected by `User B` and `User C`, requires `Internet`, which is removed by `User A`. And finally, just one video provider should be specified, but `User B` selects `Provider B` while `User C` selects `Provider A`. Our goal here is to accomplish a valid configuration which maximises the satisfaction of all these users.

IV. THE SOLUTION

The solution we propose is interpreting multi-user configuration process as a game theoretic problem. In this section, we introduce the fundamentals of game theory, point the similarities between multi-user configuration and game theory, and present our tradeoff-based bargaining approach.

A. Game theory in a nutshell

Game theory is a discipline between economics and maths, which is defined as "*the study of mathematical models of conflict and cooperation between intelligent rational decision-makers*" [8]. It has been successfully applied to many fields, like economics, political science, psychology or informatics. Generally, a game theoretic problem is a 3 elements tuple

$$\Gamma = (N, C_i, U_i), i \in N$$

where N is the set of players, C_i the set of possible strategies of each player, and U_i the utility function which measures each player satisfaction. The ultimate goal of every player is maximizing their own benefit.

Game theoretic problems can be categorized by several dimensions, depending on the rules, possible strategies or available information among other factors. For our interests, we emphasize four possible classifications by: number of players, cooperativeness, concurrency and available information.

- *1-player* vs *n-player* games: in general, game theory deals with a multiple persons game, since these games may present conflicts among players. However, there exist also single player games, where the player should obtain the maximum benefit respecting game rules.
- *Non-cooperative* vs *cooperative* games: a game is cooperative if the players are able to form binding commitments, and non-cooperative in another case. Non cooperative games consider the assumption that every player chooses the best strategy for them and also assumes that the rest of players do the same without considering alliances. It leads players to choose non-optimal strategies as a result.
- *Simultaneous* vs *sequential* games: simultaneous games are games where players are not aware about other previous players decisions, so all the decisions of every player are applied concurrently. In sequential games, players have knowledge about decisions made earlier. For instance, tic-tac-toe is a sequential game, while rock-paper-scissors is simultaneous.
- *Complete information* vs *incomplete information* game: in complete information games, every player knows the strategies and payoffs available to other players, while in incomplete information games such information is hidden.

These classifications reveal similarities with the configuration dimensions shown in Section III (Table I).Due to this similarities we state the hypothesis that multi-user configurations can be defined in terms of a game theoretic problem.

TABLE I

SIMILARITIES BETWEEN MULTI-USER CONFIGURATION PROCESS AND GAME THEORY

	Multi-user config	Game theory
Players concurrency	Mutual exclusion	N 1-person games
	Concurrent	N-person game
Player decisions	Transactional	Simultaneous
	Interactive	Sequential
Information	Customized view	Incomplete information
	Whole view	Complete information
Cooperativeness	Fixed decisions	Non-cooperative
	Weighted preferences	Cooperative

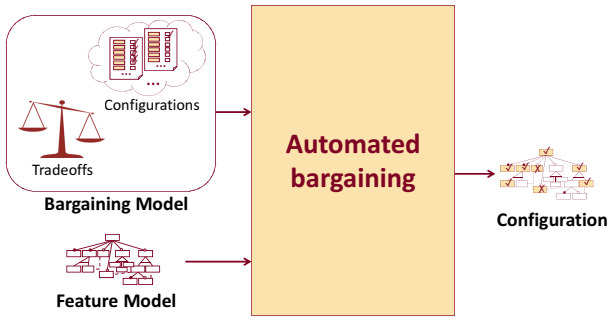

Fig. 4. Adding a bargaining to automate multi-user configuration

B. Multi-user configuration is a game theoretic problem

We consider that current multi-user configuration approaches have a weak point to deal with conflicts: they only take a variability model and a set of configurations as inputs. If any conflict arises, we have no information about how a fix could affect user satisfaction. For this purpose, we propose to add a new input (Figure 4): a set of tradeoffs for each user, where they specify alternative decisions in case of conflicts, and the impact over their satisfaction.

Taking into account tradeoffs as a new input, we interpret multi-user configurations as a game theoretic problem. Both of them involve several players, have game rules and different strategies, and all players want to maximise their own satisfaction. We model multi-user configuration as a game theoretic problem in the following way:

$$\Gamma_{conf} = (N, FM, C_i, U_i), i \in N$$

where N is the set of users who take part in the configuration process, FM is the Feature Model, C_i are the different decisions available for the users, and U_i is the user satisfaction function, defined by the tradeoffs.

C. A tradeoff-based bargaining approach

The most natural way to solve conflicts among several parties is a negotiation. Cooperative game theory, in contrast to non-cooperative, is the branch of game theory which analyses the conflicts of decision-makers from a negotiative point of view. It is proved that, despite cooperation is not always possible, it brings more satisfactory solutions for players than non-cooperative approaches [6]. The mechanism that enables the cooperation is a bargaining process [9]. This bargaining process can be performed freely by the users. However, if we want to ensure the fairness (players should be treated symmetrically) and efficiency (it is not possible to obtain any further improvement without making at least one individual worse off), an impartial arbitrator should guide the bargaining [8]. We define the bargaining process in multi-user configuration as the negotiation between the different users to achieve a valid, equitable and efficient configuration. The elements of this negotiation, i.e. the strategies C_i of the game, are tradeoffs.

A tradeoff can be defined as a compensation or compromise in exchange of something. In our case, we consider a tradeoff as giving up a decision in exchange for a compensation. Formally, we define a tradeoff j for a user i as a 3-tuple:

$$tradeoff_{ij} = (D_{ijLose}, D_{ijWin}, t_{ij})$$

D_{ijLose} is a set which represents one or more clauses the user is willing to give up. D_{ijWin} is a set which contains zero or more clauses the user obtains if the tradeoff is performed. And t_{ij} represents the impact in the user satisfaction U_i. A negative value diminishes user satisfaction, while a positive value improves it. Users have to define their tradeoffs after conflicts detection, and prior to the bargaining process. In order to ensure that users do not force unfair bargaining decisions, we establish limits for each tradeoff $t_{ij} \in [0, K]$ and user $\sum_j t_{ij} \leq L, \forall i \in N$.

Tradeoffs widen and weight the set of acceptable configurations for a user. Initially, the space of acceptable configurations for a user is S_{i0}. Conflicts appear whenever the intersection of S_0 sets is empty. It means that no configuration is acceptable for all the users. When a user defines a set of tradeoffs, the S_{i0} set widens until S_{iT}. In this way, our bargaining process goes after making the intersection of acceptable configurations non-empty, by means of enlarging such sets using tradeoffs. But the more tradeoffs are applied, the less satisfaction users perceive. So we look for the intersection of S_T sets which brings the maximum satisfaction for all the users.

D. Automating the bargaining

In order to automate the bargaining, we need to define what an impartial arbitrator is for us, and its properties. As we previously presented, such arbitrator should deal fairly and efficiently with the users. Besides, this arbitrator should know all the inputs in the process: the FM, and configurations and tradeoffs of each user. An automated behaviour also requires that users are not aware about other user decisions unless the arbitrator communicates them. In game theoretic terms, this means that the arbitrator considers a simultaneous and complete information cooperative game of N players (Table I).

The way we propose to automate the arbitrator is by means of an *Automated Analysis of Feature Models (AAFM)* operation. AAFM is a discipline which extracts valuable information from FMs in an automated way, translating the problem to logical solvers [1]. Until now, more than 30 operations have been identified, but none of them

TABLE II
SMART HOME SCENARIO TRADEOFFS

Tradeoff	Lose	Win	Value
User A 1	MoviePlayers	-	-5
User A 2	¬Internet	Ethernet	-2
User B 1	¬MoviePlayers	¬HDTV42	-2
User B 2	Video on demand	Digital Media Server	-1
User B 3	Provider B	-	-2
User C 1	Video on demand	Digital Media Server	-1
User C 2	Provider A	-	-2

tackles conflicts in multi-user configurations. The operation we propose should receive the inputs of Figure 4: a FM, and a configuration and a set of tradeoffs for each user. The output should be an equitable and efficient valid configuration.

We use a well-known criterion in the literature, the Nash bargaining solution [8, 9], to achieve an equitable and efficient output. Nash's proposal consists on maximizing the product of satisfaction increasing in a cooperative approach respect to the highest satisfaction in a non-cooperative approach. It means, $Max \prod(x_i - v_i), i \in N$, where x_i is the cooperative approach satisfaction for user i and v_i the competitive approach satisfaction for user such user. For our approach, we consider that v_i is 0 (the configuration is invalid and no agreement is reached), and cooperative approach satisfaction is u_i. So the goal function to maximise is $\prod u_i, i \in N$.

We have implemented a prototype for our automated bargaining proposal as an AAFM operation using *FaMa* [1], a Java framework for the AAFM. FaMa provides different AAFM operations [1]. For our purposes, we have extended FaMa, adding a Constraint Satisfaction Problem based implementation of this new operation using Choco solver[2]. For this implementation, the tradeoffs can be defined just over features, and in the same way we have presented them in Section IV-C. They have one or more lose clauses (D_{ijlose}), referred to already done selections or removals; zero or more win clauses (D_{ijwin}), where the user specifies new decisions; and an integer value (t_{ij}), which determines the tradeoff impact in user satisfaction.

E. Revisiting the case study

Now, we return to the case study in Figure 3 to apply our approach, using the prototype presented in Section IV-D. This scenario presents three conflicts: (*i*) a contradiction about *Movie players*, (*ii*) a conflict between *Internet* and *Video on demand*, (*iii*) and another conflict between *Provider A* and *Provider B*. Table II shows the set of tradeoffs defined by the users after the conflicts have been detected.

[1]http://www.isa.us.es/fama/

[2]www.emn.fr/z-info/choco-solver/

The conflicts are solved as follows. *Conflict (i)* is solved selecting `Movie players` and removing `HDTV42`, since `User B` lose value is lesser than `User A` value. *Conflict (ii)* involves all the three users. We must attend to Internet and Video on demand tradeoffs (`User A 2`, `User B 2`, `User C 1`), but also to provider tradeoffs since provider features are children of Video on demand. Total lose value for `User A` is lesser than the related to `User B` and `User C`, so `Internet` is selected through `Ethernet` feature. Finally, we should decide about video on demand provider for *Conflict (iii)*. Although both tradeoffs (`User B 3` and `User C 2`) have the same lose value, `User B` has been previously damaged by another tradeoff while `User C` has not. Therefore, Nash bargaining solution selects `Provider B` and discards `Provider A`. Considering M constant as the maximum satisfaction for each user, user satisfaction is: `User A` = $M-2$, `User B` = $M-2$, `User C` = $M-2$.

F. Discussion

Several aspects of our approach may lead to a discussion. Firstly, we focus our work on single-view variability models. Many large systems use different views/models for each user. However, such views refer to the same original model, so we state that our approach can be applied independently of the different views. Secondly, some doubts might arise about tradeoffs setting and bargaining convergence. Besides t_{ij} limits of Section IV-C, we propose several tradeoff preconditions to ensure that users do not force bargaining decisions: *a*) each user should define at least one tradeoff for each conflictive decision in which she is involved, *b*) win clauses (D_{ijWin}) cannot refer to conflictive decisions, and *c*) the tradeoffs of each user cannot lead to invalid configurations. Anyway, and to ensure the bargaining convergence to a valid configuration, we assign a satisfaction $u_i = 0$ for invalid configurations, and $u_i = 1$ for configurations out of S_{iT} set. In this way, any valid configuration is preferred instead of an invalid one.

V. Related work

Requirements negotiation is a classic topic in software engineering. Karlsson and Ryan [5] present requirements prioritization focused on cost-value relationships. Boehm et al. [2] efforts are focused on win-win clauses. Ruiz-Cortés et al. [12] work is more related to our approach, proposing win-lose clauses for requirements and SLAs [13] in a similar way as we do.

Several works have proposed game theory to assist services configuration, both for client side and resource allocation. Some of these works have been focused on bargaining approaches. Quan and Altmann [11] propose a bargaining method to fix the price of a SLA workflow broker. This work interprets the problem as a bargaining process with asymmetric impatience instead of proposing an arbitrator. Zulkernine and Martin [17] present a negotiation broker for adaptive and intelligent bilateral SLA bargaining. Contrasting to our approach, this work considers an incomplete information bargaining. However, most of works present non-cooperative approaches. Silaghi et al. [14] tackle resource allocation problem in competitive grid. Despite considering a incomplete information game, their approach can infer opponent's utility function and achieve a fair resource allocation. Wei et al. [15] present an iterative and non-cooperative game theoretical approach to tackle equilibrium lack in Cloud allocations. The same problem is dealt by Palmieri et al. [10], but by means of an agent-based approach towards a Nash equilibrium solution.

VI. Conclusions and further work

In this work, we have presented a proposal to deal with conflicts in multi-user configuration of FMs based in cooperative game theory. We have interpreted multi-user configuration as an automated bargaining process, where players are different users, the available strategies are tradeoffs, and the users payoff are their satisfaction for the resultant configuration. We have proposed a new AAFM analysis operation, which acts as an impartial arbitrator of the bargaining process and which produces an equitable and efficient valid configuration as output. Such operation has been implemented as an extension of FaMa framework.

Further work can take several directions. A deep evaluation of our approach is required. In this sense, it would be interesting involve real users in evaluation. We think that our approach could also be applied to cardinalities and non functional properties of FMs. And from an AAFM point of view, it would be intriguing to research about which new analysis operations the inclusion of tradeoffs may lead to.

Acknowledgements

This work was partially supported by the European Commission (FEDER), and the Spanish and the Andalusian R&D programmes (grants TIN2012-32273 (TAPAS) and TIC-5906 (THEOS)).

References

[1] D. Benavides, S. Segura, and A. Ruiz-Cortes. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 2010.

[2] B. Boehm, P. Bose, E. Horowitz, and M.-J. Lee. Software requirements as negotiated win conditions. In *Proceedings of the First International Conference on Requirements Engineering*, 1994.

[3] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged Configuration Using Feature Models. In *Software Product Lines Conference*, 2004.

[4] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report, 1990.

[5] J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *Software, IEEE*, 1997.

[6] D. M. Kreps, P. Milgrom, J. Roberts, and R. Wilson. Rational cooperation in the finitely repeated prisoners' dilemma. *Journal of Economic theory*, 1982.

[7] M. Mendonça, D. Cowan, W. Malyk, and T. Oliveira. Collaborative Product Configuration : Formalization and Efficient Algorithms for Dependency Analysis. *Journal of Software*, 2008.

[8] R. B. Myerson. *Game theory: analysis of conflict*. Harvard University Press, 1991.

[9] J. F. Nash. The bargaining problem. *Econometrica: Journal of the Econometric Society*, 1950.

[10] F. Palmieri, L. Buonanno, S. Venticinque, R. Aversa, and B. D. Martino. A distributed scheduling framework based on selfish autonomous agents for federated cloud environments. *Future Generation Computer Systems*, 2013.

[11] D. M. Quan and J. Altmann. Bilateral bargaining game and fuzzy logic in the system handling sla-based workflow. In *22nd International Conference on Advanced Information Networking and Applications-Workshops*, 2008.

[12] A. Ruiz-Cortés, R. Corchuelo, A. Duran, and M. Toro. Automated support for quality requirements in web-service-based systems. In *The Eighth IEEE Workshop on Future Trends of Distributed Computing Systems*, 2001.

[13] A. Ruiz-Cortés, A. Durán, R. Corchuelo, and M. Toro. Using Constraint Programming for the Automatic Detection of Conflicts in Quality Requirements. In *WER*, 2002.

[14] G. C. Silaghi, L. D. Şerban, and C. M. Litan. A time-constrained SLA negotiation strategy in competitive computational grids. *Future Generation Computer Systems*, 2012.

[15] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, 2010.

[16] J. White, D. Benavides, D. C. Schmidt, P. Trinidad, B. Dougherty, and Ruiz-Cortes. Automated diagnosis of feature model configurations. *Journal of Systems and Software*, 2010.

[17] F. H. Zulkernine and P. Martin. An adaptive and intelligent sla negotiation system for web services. *IEEE Transactions on Services Computing*, 2011.