2013 IEEE International Conference on Software Maintenance

WSDarwin: A Decision-Support Tool for Web-Service Evolution

Marios Fokaefs and Eleni Stroulia
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
Email: {fokaefs,stroulia}@ualberta.ca

***Abstract*—Service-oriented systems are fundamentally distributed in nature, relying on external services accessible through their public interfaces. Distributed ownership and lack of implementation transparency imply special challenges in the evolution of such systems. In order to alleviate the challenge faced by the consumers of their services, providers should, in principle, take into account the impact that service changes may have on the client applications, in addition to considering the potential benefits to be gained from the evolution of these services. In this paper, we present a decision tree to support the provider's service-evolution decision-making process. Using game theory, we construct the tree that makes explicit the value-cost trade-offs involved in considering the potential evolution of services.**

I. Introduction

The service-oriented system paradigm and associated technologies were conceived to support the reuse of functionality in the context of the development of large-scale distributed systems. Since services are consumed through standard public specifications (primarily in XML), service systems are technology agnostic and do not require any knowledge about implementation details of the services they rely upon. Although these properties have led to the easy development of component-based applications, they have also given rise to new challenges. One of them concerns the evolution of such systems. Due to the lack of implementation information between the two parties, i.e., the provider and the client, the latter is not in a position to reason about the changes of the service or the motivation behind the change. This can potentially increase the cost of adaptation for client applications and can potentially motivate the client to abandon the current provider.

When changing a service, a provider aims at maximizing some anticipated benefits, originating from the improvement of the service functionality or the extension of its features in order to acquire more revenue from clients, while, at the same time, minimizing the costs associated with actually implementing the change. These benefits and costs are direct and visible to the provider. On the other hand, there are indirect benefits and costs (also known in economic theory as *externalities* [1]) for the provider stemming from the client's reaction to the change. If the change improves the service and its quality, and meets better (or more of) the clients' needs, then they may become more committed to the service and they may be willing to pay more for it, thus generating additional revenue for the provider. On the other hand, if the clients' cost to adapt to the provider's evolved service far exceeds the benefits from the new version or there are other alternatives in the market that better suit their requirements, they may decide to switch to a competitor service, depriving the current provider from a source of income. The implication is that service providers, while being in principle self-interested, should also consider their clientèle, since they would risk losing clients and/or failing to attract new ones. When calculating the implications of a particular service change, the provider must also consider the properties of the ecosystem (providers-services-clients) and work towards the mutual benefit of all involved parties.

In this work, we extend our previous work [2], where we proposed a provider-client game to analyse their interactions during service evolution. In this paper, we propose a decision tree as a decision support tool, based on the provider-client game. We, first, revise the game and outline how it is changed from the previous version to allow for a more realistic ecosystem with potentially many providers and many clients. We also revise the best-response analysis to define the conditions, under which certain decisions are made and which are used as the decision nodes in the proposed tree. The resulting decision tree not only allows providers to make optimal decisions, but also to understand why these decisions are optimal for them and for the ecosystem as a whole. The tool is currently being built as part of WSDarwin, our web-service evolution platform [3].

The rest of the paper is organized as follows. Section II describes the proposed framework. In Section III, we provide an overview of this work's background by presenting a selection of related papers. Finally, Section IV concludes the paper.

II. The Decision Support Framework

A. The Provider-Client Game

In our previous work [2], we argued that the provider-client relationship involves conflicting interests and contradicting goals. For this reason, we proposed a provider-client game to capture this relationship. Table I shows the variables used in the game and their description. We denote provider specific variables with the superscript P and client-specific variables with the superscript C. We denote the current provider with the subscript i and the competitor with the subscript j. The differential values with subscript ei refer to the different value/price of the old version of the current service and the

1063-6773/13 $26.00 © 2013 IEEE
DOI 10.1109/ICSM.2013.68

new version. The differential values with subscript ej refer to the different value/price of the current service (old or new) and the competitive service.

TABLE I: The variables and their definitions as used in the provider-client game.

Variables	Definition
V_{ei}^{C}	the differential value of the service of the current provider i
p_{oi}	the original price the client pays for the service before the evolution
C_e	the cost of the evolution process
$C_{ai\|j}$	the cost of adaptation to the new version of the current provider's service i or the competitive service j for the client application
$p_{ei}^{E\|S}$	the price differential for the updated software when the provider just evolves (E) or when the provider also supports the client's adaptation (S)
p_{ej}	the price differential between the current provider and the competitor
V_j^C	the the value of the competitor provider j's software for the client
a	the subsidy rate, which determines what portion of the adaptation costs the provider will cover
b	the final portion of the adaptation costs that remains for the client after the provider's support

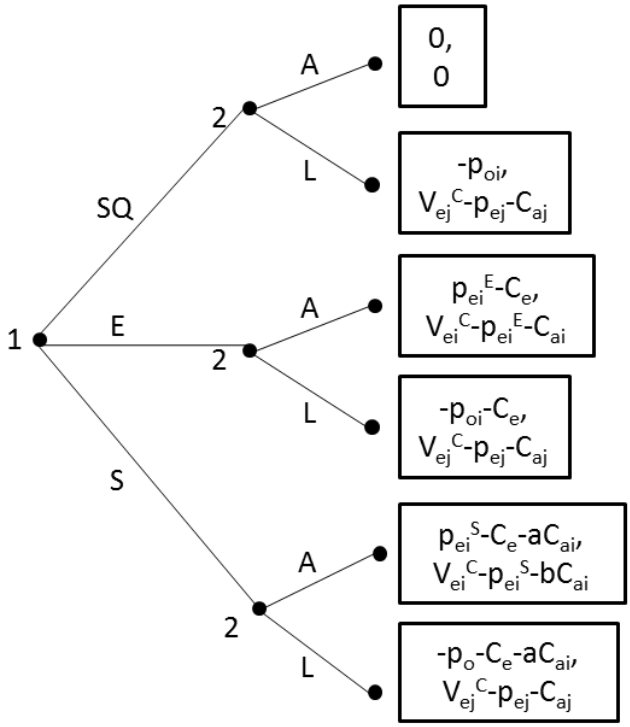

Fig. 1: The provider-client game in extensive form with the payoffs as the values for the leaves.

Figure 1 shows the provider-client game in its extensive form. The two values in the leaves of the tree correspond to the payoffs for the provider (first value) and the client (second value). In the game, the provider plays first and the possible actions are to:

1) **Maintain the status quo** (SQ) of the service and make no change. In this case, there is no difference in the provider's payoff, if the client decides to stay, but the provider loses the original income p_{oi}, if the client decides to leave.
2) **Evolve the service** (E), gain the increased revenues p_{ei}^{E}, if the client stays or lose the original income p_{oi}, if the client leaves and incur the evolution costs C_e.
3) **Support the client's adaptation (S)** and evolve the service. In this case, the provider gains the increased revenues p_{ei}^{S}, if the client stays or loses the original income p_{oi}, if the client leaves and incur the evolution C_e and part of the adaptation costs aC_{ai}.

Then the client responds and the possible actions are to:

1) **Adapt to the change (A)**, gain the differential of the value of the service from the old version to the new V_{ei}^{C} (or zero if the provider doesn't evolve the service) and incur the price differential (zero, if the provider doesn't evolve the service, p_{ei}^{E}, if the provider evolves or p_{ei}^{S}, if the provider evolves and supports) and any adaptation costs (zero, if the provider doesn't evolve the service, C_{ai}, if the provider evolves or bC_{ai}, if the provider evolves and supports, where $C_{ai} > bC_{ai}$).
2) **Leave the current provider (L)**, gain the differential of the value of service between the current provider and the competitor V_{ei}^{C} and incur the price differential p_{ej} and any adaptation costs C_{aj}.

The reason why we permit only two alternatives for the client is due to the nature of service systems. Unlike other modular software, for which clients can have a local copy of the software module and invoke on demand, in service systems, the web service has to be always online, since it is accessible over a network. Any change in the service may disrupt the proper function of the clients. Maintaining multiple versions of the service at the same time may prove very costly for the provider. In this case, the provider has two options. The first is to give to the clients a grace period before making the changes to the service, so that they have time to react properly to the change. An example of this case is Twitter [1], which released API v1.1 in September 2012 as a replacement for the old v1, which was completely retired in March 2013. At the end of the grace period, the clients will still have to decide whether they will adapt to the new version or switch providers. The second option is that the provider creates adapters that have the old version's interface but invoke the new version of the service. This idea was proposed and investigated by Kaminski et al. [4] and Fokaefs and Stroulia [3]. This option is in fact modelled by the support action of the provider in the proposed game.

A first difference between this game and its previous version [2] is that we no longer consider a value of the service for the provider and the income is only represented by the price.

[1] https://dev.twitter.com/blog/planning-for-api-v1-retirement (last accessed 17 June 2013

Another very important difference is that the client can now leave the current provider even if the latter retains the status quo. Assuming rationality from the client's part, if no provider evolves their services, then the client has no reason to switch providers. However, if one of them evolves the service, the client may opt to leave the current provider, if a competitor offers a better service. For simplicity purposes, we do not include the competitors as strategic players in the game, but we rather include the effect of their decisions in the calculations of the payoffs of the client.

Our analysis can be easily extended for multiple clients. We can safely argue that a client's decision does not depend on the other clients' decisions. Therefore, the outcomes and the payoffs for all players depend only on the provider's decisions and the competition. Therefore, we can run an instance of the game for each client and then aggregate the results and make the decisions that satisfies as many clients as possible.

B. Solution Concepts and the Decision Tree

Having defined the interactions between the provider and the client, we can now proceed to analyse the game using the backward induction algorithm and the method of best-response analysis, i.e., what action a player prefers given the preferences of the other players. Backward induction is a solution algorithm for sequential games, which first calculates the optimal decision for the player that plays last for each of the previous player's actions. Then these optimal actions are taken as a given for the previous player. The process is continued until we reach the player that plays first. The final path gives us the equilibrium of the game. The best-response analysis will give us the conditions under which a player has certain preferences and which we will use to construct the decision nodes of the decision tree.

Table II presents the best-response analysis for the provider-client game. As we can see from the table, conditions 2, 4 and 6 (i.e., when its more preferable for the client to leave the current provider) will never hold. This means that if the client leaves (i.e., conditions 7 and 8 do not hold), the provider will opt to retain the status quo of the service since this is the action that minimizes the losses for the provider. In an expanded ecosystem with many clients, the provider may balance the losses from clients that leave by attracting potentially new clients.

Figure 2 shows the final decision tree for the provider when considering the evolution of a service. The decision nodes are shown as rectangles with the number of the corresponding conditions. For each of the decisions we have two edges; one when the conditions is **T**rue in the left and another when the condition is **F**alse in the right. The end nodes of the tree representing the outcome of the decision process are shown as solid triangles with the final action for the provider.

As discussed above, all the paths for which the client leaves (i.e., $1(T) -> 5(T) \rightarrow 3(T) \rightarrow 8(F)$, $1(T) \rightarrow 5(F) \rightarrow 7(F)$ and $1(F) \rightarrow 5(T) \rightarrow 3(T) \rightarrow 8(F)$) lead the provider to retain the status quo of the service. The provider is led to the same outcome in two more paths ($1(F) \rightarrow 5(F)$ and $1(F) \rightarrow 5(T) \rightarrow 3(F)$, in which, independently of the client's decision, it is not in the interest of the provider to evolve since the costs exceed any potential income. For the other outcomes, one path leads to the evolution of the service ($1(T) \rightarrow 5(F) \rightarrow 7(T)$). The subpath $5(T) \rightarrow 3(T) \rightarrow 8(T)$ leads to the evolution of the service with support to the client regardless of whether condition 1 holds or not. In the leftmost subtree under the root, condition 3 can never be false, since, because conditions 1 and 5 are true, we have that $S \succ E \succ SQ$. Furthermore, under the same subtree, when condition 5 is false, we have that $E \succ SQ$ and $E \succ S$, which means that action E is a dominant strategy for the provider and we don't have to check condition 3.

TABLE II: Best-response analysis for the provider-client game.

		When the client prefers...	
		$A \succ L$	$L \succ A$
Provider	$E \succ SQ$	1. $p^E_{ei} > C_e$	2. $-C_e > 0$
	$S \succ SQ$	3. $p^S_{ei} > C_e + aC_{ai}$	4. $-C_e - aC_{ai} > 0$
	$S \succ E$	5. $aC_{ai} < p^S_{ei} - p^E_{ei}$	6. $-aC_{ai} > 0$
		When the provider prefers...	
		$E \succ S$	$S \succ E$
Client	$A \succ L$	7. $V^C_{ei} - V^C_{ej} > p^E_{ei} - p_{ej} + C_{ai} - C_{aj}$	8. $V^C_{ei} - V^C_{ej} > p^S_{ei} - p_{ej} + bC_{ai} - C_{aj}$

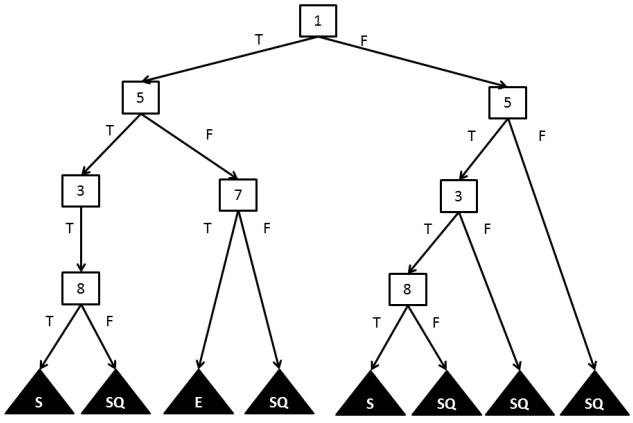

Fig. 2: The evolution decision tree for the service provider.

This decision tree is a simple tool that requires a considerate amount of information to be useful. However, this information is reasonably easy to obtain and in fact good enough estimates can be produced for all the variables required. First of all, there are variables that can be manipulated to guide the provider's decision towards a desired outcome. The provider can increase the value of the service at will and offer it at a competitive price, thus effectively overcoming competition. Furthermore, the provider can decide on the level of support

to the client to ease the adaptation process giving the client the incentive to stay. We can estimate the rest of the variables either by using formal models (i.e., for costs) or by surveying the environment of the service (i.e., for values and prices of competitive services).

III. Software Engineering Economics Background

Although software economics is a relatively mature field, analysing the cost and value of a particular software-engineering activity is an ever challenging problem. This problem is exacerbated in the case of service systems, because of the peculiarities of such systems, some of which we have highlighted in this work. In their work, Boehm and Sullivan [6], [7] outline these challenges and also how software-economics principles can be applied to improve software design, development and evolution.

Boehm and Sullivan define software engineering fundamentally as an activity of making decisions over time with limited resources, and usually in the face of significant uncertainties. *Uncertainties* pose a crucial challenge in software development that can lead to failure of systems. Uncertainties can arise from inaccurate estimation. For example, cost-estimation techniques and models that used to work for traditional development processes may not apply directly to modern architecture styles and development processes, such as web services. Furthermore, due to lack (or inadequacy) of economic and business information software projects may be at risk. They recognize the need of including the *value added* from any design or evolution decision. However, as they point out usually there are no explicit links between technical issues and value creation. It is critical to understand that the value added by evolving a system does not only depend on technical success but also on market conditions. It is stressed that the cost should not be judged in isolation. As Parnas suggests "for a system to create value, the cost of an increment should be proportional to the benefits delivered" [8]. Finally, the authors claim that there is a need for not only better cost estimation models but also stronger techniques for analysing benefits.

The provider-client game as presented in this work is a clear example of an ecosystem where externalities exist. An externality is an indirect cost or benefit of consumption or production activity, in other words, effects on agents other than the originator of such activity which do not work through the price system [1]. External effects such as these can lead to suboptimal, or inefficient outcomes, for the system as a whole, whereby both parties by acting independently end up less well off than they could do if they coordinated their actions or if the decision maker (in this case the provider) took into account the external effects of any action.

The Coase theorem [5] argues that an efficient outcome can be achieved through negotiations and further payments between the involved parties under certain conditions (the parties act rationally, transactions costs are minimal, and property rights are well-defined). In this work, the relationship between the provider and the client as we have described it through the game is an example of the Coase theorem.

IV. Conclusion and Future Plans

We argue that evolution in service-oriented architectures is a complicated task due to the complex dependencies between the participants of the service ecosystem. These dependencies are not only of technical nature but they also have economic and business implications. As a result strategic decisions concerning the evolution of a service should consider both technical and business dimensions of the ecosystem. In this paper, we showed how Game Theory can be used to model these dependencies and interactions between a service provider and a service client. We also proposed a decision tree as a tool to support the provider's decision-making process. This tool is based on estimates of economic parameters such as values, costs and prices and takes into account the clients' reactions to providers' decisions while at the same time considering the competition.

Although the model presented in this paper focuses on a single provider and a single client, it can be easily extended for more complicated ecosystems, while the general idea remains the same; service evolution in the presence of externalities. To this end, we plan to create decision trees for providers in different market settings: few clients with many providers (*oligopsony*), few providers with many clients (*oligopoly*), or many providers with many clients (*free/competitive market*). Another important part of our future plans is the validation of the decision support system. This can be achieved by simulating various evolution scenarios and testing our decision tree with (pseudo)randomly generated values. An alternative is an on-site evaluation of the decision tree by real service providers when considering the evolution of their service.

References

[1] J. Laffont, "externalities," in *The New Palgrave Dictionary of Economics*, S. N. Durlauf and L. E. Blume, Eds. Basingstoke: Palgrave Macmillan, 2008.

[2] M. Fokaefs, E. Stroulia, and P. R. Messinger, "Software Evolution in the Presence of Externalities: A Game-Theoretic Approach," in *Economics-Driven Software Architecture*, I. Mistrik, R. Bahsoon, R. Kazman, K. Sullivan, and Y. Zhang, Eds. Elsevier, 2013.

[3] M. Fokaefs and E. Stroulia, "Wsdarwin: Automatic web service client adaptation," in *CASCON '12*, 2012.

[4] P. Kaminski, M. Litoiu, and H. Müller, "A design technique for evolving web services," in *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research - CASCON '06*. New York, New York, USA: ACM Press, Oct. 2006, p. 23.

[5] R. H. Coase, "The Problem of Social Cost," *Journal of Law and Economics*, vol. 3, pp. 1–44, 1960.

[6] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall, 1981.

[7] B. Boehm and K. Sullivan, "Software economics: status and prospects," *Information and Software Technology*, vol. 41, no. 14, pp. 937–946, 1999.

[8] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053–1058, Dec. 1972.