Quality Attribute Game: A Game Theory Based Technique for Software Architecture Design

Kiran Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur, India
vkirankr@iitk.ac.in

Prabhakar TV
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur, India
tvp@iitk.ac.in

ABSTRACT
Game Theory is the study of multi-person behavior where the decision of one player affects the decisions of others; concepts such as Nash Equilibrium, Pareto optimality etc. analyze the optimal strategies of the players in a particular situation. In Software Architecture Design, various Quality Attributes (QAs) affect each other and the designer makes decisions based on the effectiveness of an architecture solution (AS) at the given scenario, its side-effects on other QAs, and priorities of QAs. One way to automate the architecture design process is to model the conflicting behavior of QAs as a game-theoretic problem and apply the concepts in game theory for optimal design decision analysis.

1. INTRODUCTION
From the decision view perspective, the architecture of a system is a set of design decisions [4]; for a particular scenario, a design decision (DD) is the optimal design alternative (DA). In value terms, a DD is the DA which has maximum utility/value [2]. An immediate consequence of conflict among QAs is: an AS in addition to achieving one or more QRs also imposes one or more QRs as side-effects to the system whose importance may or may not be significant. For example, Encryption which achieves data confidentiality imposes significant performance overhead in general. When side-effect based design process such as Attribute Driven Design (ADD) [1] is used, the designer makes decisions based on the effectiveness of an AS at the given scenario, its side-effects on other QAs, and priorities of QAs. We propose a Quality Attribute Game (QAG) technique which enables the automation in exploring the optimum DDs based on QR values. In short, QAG technique is described as follows: the dependency nature of the QAs is modeled as a game-theoretic problem and the design decisions (optimal DAs) are analyzed using game theory concepts.

2. Game theory solution concepts.
2.1 Nash equilibrium (NE).
Definition: A strategy profile (s1, s2, .., sn) is said to be in NE if every player is playing his best strategy to the other players' best strategies.

Figure 1 illustrates the pivoting algorithm for the Prisoners' Dilemma problem [3, 5] where strategy profile (Fink, Fink) is in NE. Pivoting is done as follows:
Player1's pivot w.r.t. column 1 = max (-1, 0) = 0
Player1's pivot w.r.t. column 2 = max (-9, -6) = -6
Player2's pivot w.r.t. row 1 = max (-1, 0) = 0
Player2's pivot w.r.t. row 2 = max (-9, -6) = -6

2.2 Pareto Optimality (PO).
Definition: The strategy profile S’ - (s’1, s’2, .., s’n) is said to be PO than strategy profile S - (s1, s2, .., sn), if payoff of a player can be increased by changing S to S’ without decreasing payoff of other players.

Example: Suppose S = (6, 7) and S’ = (6, 9), S’ is PO than S. If payoffs of S = (7, 6) and S’= (8, 5) then S’ is not PO than S. where as payoffs of S = (7, 5) and S’= (8, 7) then S’ is PO than S.

2.3 Backward-induction Equilibrium (BIE).
This solution concept follows bottom-up approach to solve sequential games [3, 5] where players take their move according to the specified sequence among the players.

Figure 2 illustrates the pivoting algorithm [3, 5] for a sample sequential game where strategy profile (s2, s2) is in BIE. Pivoting is done as follows:
Player2's pivot w.r.t. player1’s s1 = max (5, 0) = 5
Player2’s pivot w.r.t. player1’s s2 = max (-2, 4) = 4
Player1’s pivot w.r.t. player2 move = max (4, 5) = 5

Figure 2: BIE for a sequential game.
3. Quality Attribute Game.

Figure 3 shows the components and working of the QAG technique. Following subsections discuss each component with an illustrative example.

![Figure 3: Components of Quality Attribute Game.](image)

3.1 Architecture Knowledge.

This component maintains the information of different QAs, QRs, ASs, and relationships among them. Figure 4 illustrates a sample knowledge base which will be used in the illustrative example.

![Figure 4: Architecture Knowledge Base.](image)

3.2 Design Alternative Explorer.

This component is responsible for the following three tasks in ADD process: Architecture Drivers (ADs) determination, DA analysis, and Side-effects analysis. From Table 1, Authenticate user, and Reduce response time are ADs (max-valued QRs) for Security and Performance QAs. For these ADs, DAs and their side-effects can be referred from Figure 4. In Table 1, either side-effect or its corresponding QR but not both is shown whichever improves understandability from user perspective.

<table>
<thead>
<tr>
<th></th>
<th>Value assigned QRs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Authenticate user.</td>
</tr>
<tr>
<td>5</td>
<td>Reduce Response Time.</td>
</tr>
<tr>
<td>4</td>
<td>Reduce network transactions.</td>
</tr>
<tr>
<td>3</td>
<td>Reduce risk of local hackers.</td>
</tr>
<tr>
<td>2</td>
<td>Insufficient Resources at client-side.</td>
</tr>
</tbody>
</table>

3.3 Design Alternative Value Calculator.

Due to the conflicting behavior of QAs the value of an AS cannot be computed independently, but should be calculated with respect to its conflicting AS. If AS1 and AS2 are two conflicting ASs which satisfy the ADs QR1 and QR2 respectively and impose side-effects SD1 and SD2 respectively, the value calculation is expressed as formula (2) based on formula (1).

\[
\text{Benefit}(\text{AS1}) = \text{Initial-benefit}(\text{AS1}) + \text{Extra-benefit}(\text{AS1}) - \text{Imposed-loss}(\text{AS1}) .. (1)
\]

\[
\text{Value}(\text{AS1 w.r.t AS2}) = \text{Value}(QR1) + \text{Value}(SD2 if \text{AS1 achieves}) - \text{Value}(SD1 if \text{AS2 unachieved}) .. (2)
\]

Example: Value(Caching w.r.t Password) = 5 + 4 – 3 = 6.

3.4 Game Theory based optimum designer.

NE is used to analyze the optimal design decisions for each QA; the valuecomputing function shown in (2) acts as payoff function. The strategy profile (Biometrics, Caching) is in NE for the scenario shown in Table 1.

3.5 Multiple strategy profiles in NE.

NE does not result in a unique solution in all cases; in such a case, optimum among them is found using PO. BIE is applied in situations where PO is inapplicable (Ex: (7, 6), (8, 5)), but in order to apply BIE a sequential order among players must exist; we resolve this issue also using QR values - ADs is sorted in ascending order of their values and this order remains the sequence of play among the QAs.

4. Conclusion and Future work

We discussed how game theory can be applied to automate side-effect based architecture design process such as ADD; we see two major use cases of our technique - Architecture Development Environment, and Service personalization. Game theory looks promising to be able to deliver a computational solution to an architecture design problem. Substantial work needs to be done in encoding architecture knowledge and designing appropriate mechanisms for this domain.

5. References


