Aggregate Contribution of Decomposed Intentional Elements

Aprajita[1], Gunter Mussbacher[1]

[1]Department of Electrical and Computer Engineering, McGill University, Montreal, Canada
aprajita.aprajita@mail.mcgill.ca, gunter.mussbacher@mcgill.ca

Abstract. We present aggregate contribution as a solution for two long-standing issues with goal modeling languages such as i* and GRL which support decomposition and contribution links for intentional elements. An aggregate contribution summarizes the contributions of constituent parts (child elements) at the parent level, giving the modeler an overview of the actual impact of the parent while still enabling the correct evaluation of the goal model.

Keywords: Goal Modeling; Decomposition; Contribution; GRL; Goal-oriented Requirement Language; URN; User Requirements Notation; Evaluation

1 Introduction

Goal models are graphs of intentional elements such as goals and tasks connected by various relationships and optionally assigned to actors. Common relationships supported by many goal modeling notations (e.g., i* [7], the Goal-oriented Requirement Language (GRL) [3], the NFR framework [2], and KAOS [6]) are contributions and decompositions. This paper addresses shortcomings related to (a) a lack of support for relationship abstractions [5] and (b) analysis anomalies [5] in goal models that support decompositions and contributions.

In Section 2, we briefly discuss the two aforementioned issues and further motivate our proposed solution with the help of an example. Section 3 then explains how we address these two issues in detail. We use GRL to visualize the examples, but the solution applies to any other goal modeling notation with decompositions, contributions, and a propagation-based evaluation mechanism. The paper concludes and briefly discusses future work in Section 4.

2 Motivating Examples

"Sloth" (Lack of Support for Relationship Abstractions) and "Wrath" (Analysis Anomalies) are two of the eight "deadly sins" [5] that are still haunting GRL and other goal modeling languages today. Currently in GRL, it is not possible to describe the relationships of an intentional element based on the relationships of its constituent parts. As shown in Figure 1(a), the constituent parts of task A, i.e., A1 and A2, do not make their individual contributions apparent at the parent level. This may lead to

ambiguities as the parent task does not seem to be contributing to the goal. To overcome this *slothful* [5] behavior, a better visualization is needed of the total contributions of any given intentional element. Adding all contributions manually in the goal model to overcome "sloth" gives birth to "wrath". A relationship shown at more than one level of abstraction creates an analysis anomaly, because it should be taken into account only once but is actually considered multiple times. Figure 1 describes the three cases in a goal model evaluated with a quantitative evaluation algorithm [1]. As shown, (a) and (b) are equivalent (evaluation of goal G is 30), but (c) results in a different evaluation (i.e., 60) because of the repetition of the same relationship at two different abstraction levels: (i) the individual contributions of A1 and A2 to goal G and (ii) the combined contribution at the parent level.

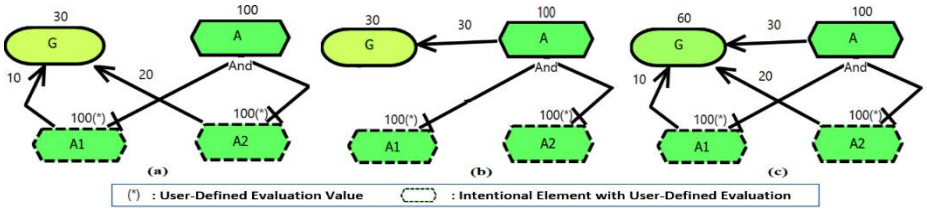


Figure 1: Evaluation of goal model with current evaluation algorithms

Similar issues exist for qualitative and mixed evaluation algorithms [1]. To tackle these issues, the concept of *aggregate contribution* for decomposed parent elements is introduced and a proof of concept implementation provided in jUCMNav [4].

3 Inferring Aggregate Contributions

When an intentional element (task A), that is decomposed into A1 and A2, contributes to another intentional element (goal G), two cases arise. First, task A does not have any individual contribution to goal G and contributes only through its constituent parts, i.e., A1 and A2. In this case, only contributions of A1 and A2 should be considered while evaluating goal G (i.e., G should evaluate to 30 in Figure 1(c)). Second, task A, along with contributing through its parts, also has its own individual contribution to goal G. Here, contributions of all elements (A, A1, A2) should be considered for the evaluation of goal G (i.e., G should evaluate to 60 in Figure 1(c)). The second case often occurs when the parent captures contributions common to all its parts.

The new concept of *aggregate contribution* clearly differentiates these two cases by showing the range of contributions for a parent element based on its parts and the decomposition type. For an AND decomposition, all parts contribute for the parent. For an OR decomposition, at least one part but possibly all parts contribute for the parent. For an XOR decomposition, exactly one part contributes for the parent. An *aggregate contribution* is a range of values, because it shows the result of all possible combinations of the contributions of its constituent parts (1 for AND, 2^N-1 for OR, and N for XOR with N being the number of parts). A new icon (**A:**) and its textual equivalent "**A:**" are used to denote a contribution link with *aggregate contribution*, followed by the actual value of the *aggregate contribution*.

All common GRL evaluation algorithms (quantitative, qualitative, mixed) first calculate for a given element its evaluation value coming from decomposition links, then contribution links, and finally dependency links [1]. *Aggregate contribution* affects the calculation of the contribution links, in that a link with only an *aggregate contribution* must not be considered by the evaluation algorithm. The *aggregate contribution* itself is calculated for each decomposed parent element before the evaluation.

Quantitative Evaluation Algorithm. This algorithm uses quantitative contribution values (e.g., task A1 contributes 10 to goal G in Figure 1(c)) and quantitative evaluations of contributing elements (e.g., 100 for task A1) to determine an element's evaluation that is coming from contribution links (e.g., for task A). An *aggregate contribution* is only calculated for a decomposed parent element that has an outgoing contribution link and is decomposed into one or more constituent parts. A modeler, therefore, has control over whether to show an *aggregate contribution* by explicitly adding the outgoing contribution link for the decomposed parent element. In addition, an *aggregate contribution* may also be indicated with a Boolean flag added to the metaclasses representing the goal model or intentional elements in the GRL metamodel. The *aggregate contribution* calculated for this algorithm is referred to as *quantitative aggregate contribution* (QNAC).

Table 1 defines the formula for calculating the QNAC values for the three decomposition types, showing the cases for only positive contributions, only negative contributions, and mixed contributions of parts. It considers that the contributions by an element's parts may consist of both individual and range contributions, both positive or negative. A range contribution may exist because a part of an element may itself be a decomposed element with an *aggregate contribution*, an example of which is shown in Figure 3(b). An *aggregate contribution* is a range represented by "**A:[Min,Max]**", except where the minimum and maximum values are the same which is simply represented by "**A:Max**".

Contributions ↓	**AND**		**OR**		**XOR**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
Positive	$\Sigma C_i + \Sigma C_{li}$	$\Sigma C_i + \Sigma C_{ri}$	$\min(Cp_i, Cp_{li})$	$\Sigma Cp_i + \Sigma Cp_{ri}$	$\min(C_i, C_{li})$	$\max(C_i, C_{ri})$
Negative			$\Sigma Cn_i + \Sigma Cn_{li}$	$\max(Cn_i, Cn_{ri})$		
Mixed			$\Sigma Cn_i + \Sigma Cn_{li}$	$\Sigma Cp_i + \Sigma Cp_{ri}$		

Legend: $\boldsymbol{C_i}$**:** Individual quantitative contribution value of part i
$\boldsymbol{C_{li}, C_{ri}}$**:** Left (minimum) and right (maximum) quantitative contribution values, respectively, of a range contributed by part i
$\boldsymbol{Cn_i}$**:** Individual quantitative contribution value of part i, when i contributes negatively
$\boldsymbol{Cp_i}$**:** Individual quantitative contribution value of part i, when i contributes positively
$\boldsymbol{Cn_{li}, Cn_{ri}}$**:** Left (minimum) and right (maximum) quantitative contribution values, respectively, of a range contributed by part i, when it contributes negatively
$\boldsymbol{Cp_{li}, Cp_{ri}}$**:** Left (minimum) and right (maximum) quantitative contribution values, respectively, of a range contributed by part i, when it contributes positively

Note: All calculated minimums and maximums are capped to -100 and 100, respectively, to respect valid GRL contribution and evaluation values.

Table 1: Aggregate contribution values for different decomposition types

Figure 2(a) shows an example goal model with an AND decomposition, where the QNAC value has been calculated for goal B contributing to goal A, resulting in "**A:20**" (Min = $\Sigma C_i + \Sigma C_{li}$ = Max = $\Sigma C_i + \Sigma C_{ri} = \Sigma C_i$ = 30 + (-10) as $\Sigma C_{li} = \Sigma C_{ri}$ = 0). Figure 2(b) shows an example goal model with an OR decomposition and parts con tributing both positively and negatively to A. Thus, the QNAC value of goal B contributing to goal A is "**A:[-10,45]**" (Min = $\Sigma Cn_i + \Sigma Cn_{li}$ = (-10); Max = $\Sigma Cp_i + \Sigma Cp_{ri}$ = 20 + 25). Figure 2(c) shows an example goal model with an XOR decomposition type. Thus, the QNAC value of goal B contributing to goal A is "**A:[-5,25]**" (Min = min(C_i, C_{li})= min (10,-5,25); Max = max(C_i, C_{ri}) = max (10,-5,25)).

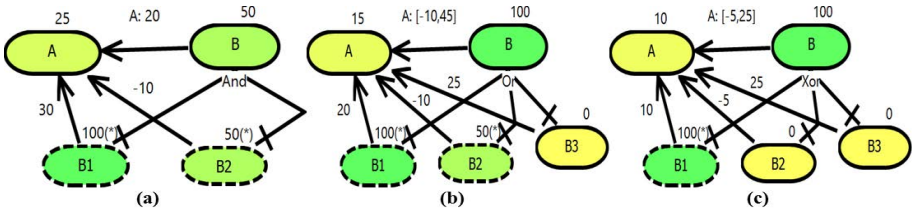


Figure 2: Quantitative aggregate contribution

Note that *aggregate contribution* as proposed here abstracts from the contributions of constituent parts only, but does not abstract from all combinations of contributions and evaluations of constituent parts. In that sense, the abstraction is based on the static goal model specification and does not consider the evaluation of the goal model. This can clearly be seen in Figure 2(a). Propagating the evaluation of B (50) to A with only the *aggregate contribution* (20) results in 10 and not the correct 25. It is possible to consider an *aggregate contribution* as an abstraction of the evaluation instead of the static goal model specification. In that case, however, AND as well as XOR decompositions have to be treated like an OR decomposition in Table 1 (i.e., the aggregate contribution in Figure 2(a) would be "**A:[-10,30]**" and in Figure 2(c) "**A:[-5,35]**"; this would ensure that the evaluation of B always falls into the range defined by the *aggregate contribution* of A on B and the evaluation of A).

As discussed earlier, a decomposed element may also contribute individually in addition to its parts. In Figure 3(a), B contributes to A individually as well, unlike the examples in Figure 2. In this case, the QNAC value for B is calculated the same way as for the examples in Figure 2 and the contribution of B is added on top. Thus, the evaluation of goal A is 40 (10 as shown in Figure 2(c) plus (30 * 100)/100 from B).

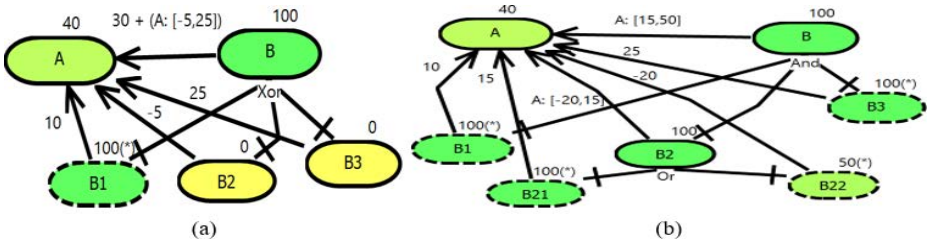


Figure 3: Individual & aggregate contributions (a) and hierarchies (b)

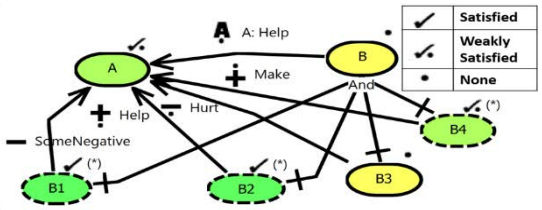
Figure 3(b) gives an example of a hierarchy of decompositions, where B is decomposed into B1, B2, and B3. B2 is then further decomposed into B21 and B22. Using bottom-up, forward propagation [1], B2's QNAC value "**A:[-20,15]**" is calculated before B's. When B's QNAC value is calculated, B2's QNAC range is taken into account resulting in "**A:[15,50]**" (Min = $\Sigma C_i + \Sigma C_{li}$ = 10 + 25 + (-20); Max = $\Sigma C_i + \Sigma C_{ri}$ = 10 + 25 + 15). The same bottom-up approach applies to decomposition hierarchies with arbitrary depths and various combinations of decomposition types.

Qualitative Evaluation Algorithm. This algorithm uses qualitative contribution values (e.g., goal B1 contributes "Some Negative" to goal A in Figure 4(b)) and qualitative evaluations of contributing elements (e.g., "Satisfied" for goal B1 as indicated by a checkmark) to determine an element's evaluation that is coming from contribution links (e.g., for goal B). In GRL, a qualitative contribution value may be one of the following discrete values [1], listed in descending order: "Make", "Some Positive", "Help", "Unknown", "Hurt", "Some Negative", and "Break". "Make", "Some Positive", and "Help" are positive contributions and opposites of the negative contributions "Break", "Some Negative", and "Hurt". "Make" and "Break" are said to be equal strength but in opposite directions, as are "Some Positive" and "Some Negative" as well as "Help" and "Hurt". The *aggregate contribution* calculated for this algorithm is referred to as *qualitative aggregate contribution* (QLAC).

For OR and XOR decomposition, the QLAC value is a range represented by "**A:[Min,Max]**", where Min is the lowest contribution among the contributing parts and Max is the highest contribution among the contributing parts. AND decompositions are handled the same way, if the purpose is to abstract from the evaluation result instead of the contributions only as discussed earlier for QNAC values.

	Hurt	**Some Negative**	**Break**
Make	Some Positive	Help	**None**
Some Positive	Help	**None**	Hurt
Help	**None**	Hurt	Some Negative

(a)



(b)

Figure 4: Contribution combinations (a), QLAC for AND decomposition (b)

To calculate the QLAC value abstracting contributions only for an AND decomposition (e.g., the QLAC value of B contributing to A in Figure 4(b)), the count of each type of qualitative contribution of the constituent parts of B to A is determined first. Then, the dominant for each strength is identified, i.e., a positive contribution and a negative contribution of the same strength (bottom-left to top-right diagonal in bold in Figure 4(a); e.g., "Make" and "Break") cancel each other out (= "None") and any remaining ones are dominant. Afterwards, the strongest contribution (most positive or most negative depending on the dominant) is combined with the strongest opposite contribution (most negative or most positive, respectively) based on Figure 4(a), until only either positive or negative contributions are left. Finally, the highest (in case only positive contributions are left) or the lowest (in case only negative contributions are

left) contribution value is selected as the QLAC value of B to A. If all contribution values cancel each other out, the QLAC value is considered to be "Unknown".

For example, Figure 4(b) shows an AND decomposition, where the parts of B contribute as follows to A: "Make", "Help", "Some Negative", and "Hurt". One "Make" and zero "Break" result in one "Make" being dominant. Zero "Some Positive" and one "Some Negative" result in one "Some Negative" being dominant. One "Help" and one "Hurt" cancel each other out. Since "Make" is the strongest remaining contribution, it is combined with the strongest opposite contribution, i.e., "Some Negative", resulting in one "Help" according to Figure 4(a) (row 2, column 3).

Similar to the quantitative evaluation algorithm, decomposition hierarchies of arbitrary depths and various combinations of decomposition types are handled with a bottom-up approach.

Mixed Evaluation Algorithm. This algorithm uses qualitative contribution values and quantitative evaluations of contributing elements to determine an element's evaluation that is coming from contribution links [1]. Consequently, *aggregate contributions* are calculated for this algorithm using the same approach as discussed for the qualitative evaluation algorithm.

4 Conclusions and Future Work

This paper discusses the issues "Sloth" (Lack of Support for Relationship Abstractions) and "Wrath" (Analysis Anomalies) in GRL and proposes the concept of *aggregate contribution* to address them. An *aggregate contribution* indicates the contributions of constituent parts for a decomposed parent element, given a sense of the overall combined contribution of the parent when all its parts are taken into account. The calculation of *aggregate contribution* for different decomposition types and evaluation algorithms is defined and detailed with the help of illustrative examples. In future work, we plan to find ways to abstract other goal modeling concepts such as dependencies and indicators for parent elements.

References

[1] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the Goal-oriented Requirement Language". *International Journal of Intelligent Systems* (IJIS), Wiley, vol. 25, no. 8, pp. 841–877, 2010.

[2] L. Chung, B.A. Nixon, E. Yu and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[3] ITU-T, *User Requirements Notation (URN) – Language definition*. ITU-T Recommendation Z.151 (11/08), Geneva, Switzerland, November 2008; http://www.itu.int/rec/T-REC-Z.151/en.

[4] jUCMNav, development build 6.0.D0610.v1626, http://jucmnav.softwareengineering.ca/jucmnav.

[5] G. Mussbacher, D. Amyot, and P. Heymans, "Eight Deadly Sins of GRL". *5th International i* Workshop (iStar 2011)*, Trento, Italy, CEUR-WS 766:2-7, 2011.

[6] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons Ltd, 2009.

[7] E. Yu, *Modeling Strategic Relationships for Process Reengineering*. Ph.D. thesis, University of Toronto, Canada, 1995.