Received October 12, 2020, accepted October 30, 2020, date of publication November 10, 2020, date of current version November 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3037210

Goal Model Evaluation Based on State-Space Representation

MOHAMED ABDEL-MONEM, ISLAM EL-MADDAH, AND HANI KAMAL MAHDI
Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University, Cairo 11517, Egypt
Corresponding author: Mohamed Abdel-Monem (mmonem@gmail.com)

ABSTRACT Goal models have been used for the last two decades in various disciplines to represent business, organizations, and individuals' objectives. Several methodologies and standards have emerged, and various goal analysis and evaluation algorithms have been introduced serving different sectors, including decision support. In contrast to most researches which are based mainly on simulation to predict the satisfaction levels of final goals, this research proposes a new framework for evaluating goal models based on the state-space representation that is used extensively in control systems. This new approach brings the theories and literature of state-space representation of systems to goal models, opening a new direction for using its available mature techniques and tools, for goal model analysis and evaluation. A hypothetical goal model, which can be used for policymaking after the COVID-19 pandemic, is presented as an example of how the proposed framework can be used, and the results that can be obtained.

INDEX TERMS Goal model, goal modeling, goal model evaluation, goal model reasoning.

I. INTRODUCTION

Since their introduction in the early 90s, goal models have been used in various disciplines, especially software requirements analysis [1]–[3], [33]. Approaches have been developed to identify possible requirement conflicts [28] and to evaluate whether particular conditions can eventually lead to the satisfaction of specific requirements [29]. They have also been used to evaluate alternative system designs [12]. Goal models have also been used for business process reengineering of pre-existing processes for achieving performance improvements [4], [23], [31], [32]. A software development methodology has also been built around goal modeling [6]. Other applications of goal models in autonomous agent-based systems can be found in [5], [25]; also, other examples for applications in decision making are described in [19], [24].

A goal can be described as a desirable condition in the world, and the goal model is a graph or a language that represents the relationship between some goals from some viewpoint. Goals may be formulated at different abstraction levels, ranging from high-level strategic concerns down to low-level technical concerns [28].

In the early 90s, while focusing entirely on software requirements engineering, KAOS (Knowledge Acquisition in autOmated Specification) was the first framework featuring goal models that gained wide popularity [1]. The framework introduced the notion of goal reduction in which a goal can be refined by other goals through AND/OR links. For an intermediate goal in the model, going up in the model can answer the question, "why do we have this goal?" while going down answers "how can we reach this goal?"

i∗ framework emerged in 1995 by Eric Siu-Kwong Yu in 1995 in his Ph.D. thesis [31]. It used the notion of the intentional element as a hypernym for goal, task, resource, and softgoal, which can be defined as a state in the world desired to be achieved without sharply defined criteria. In addition to the "how" and "why" dimensions in goal models, it introduced a new dimension, "who", and presented the intentional dependency among stakeholders in the system.

i∗ influenced NFR (Non-functional Requirements Framework) and formed a basis for other frameworks and methodologies like Tropos [6], GRL (Goal-oriented Requirements Language), and iStar 2.0 [9], which represents the core concepts of *i*∗. The evolution of the goal model approaches and frameworks are graphically illustrated in [10]. GRL is a part of the URN, which became an ITU standard for elicitation, analysis, specification, and validation of requirements [15]. Several years later, Techne [16] proposed models supporting optional goals and preferences. More elaborate discussions about goal modeling methodologies can be found in [10], [13], [17].

The associate editor coordinating the review of this manuscript and approving it for publication was Fangfei Li.

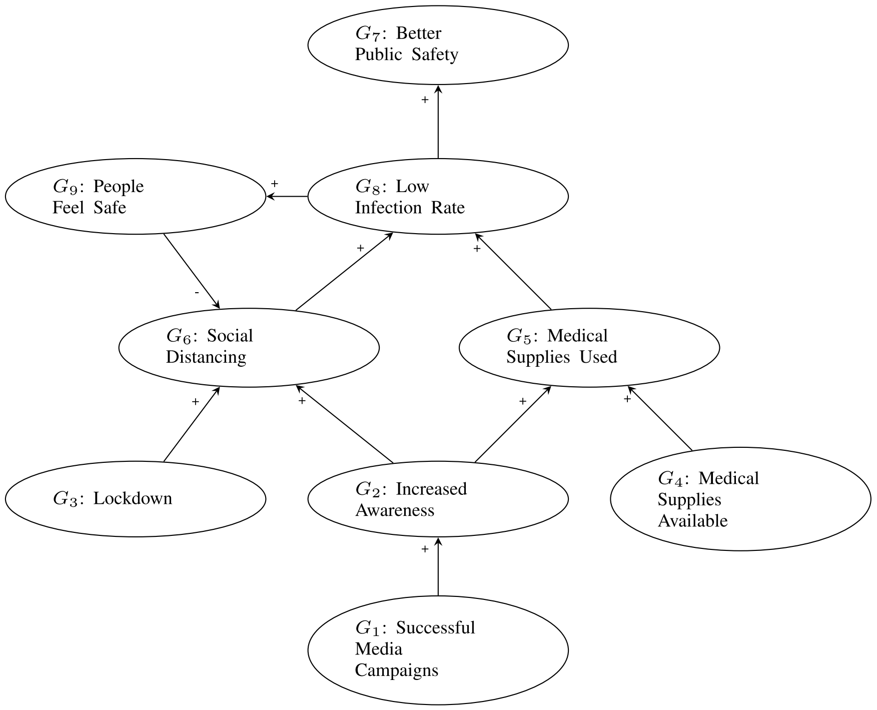

G_7: Better Public Safety
G_9: People Feel Safe
G_8: Low Infection Rate
G_6: Social Distancing
G_5: Medical Supplies Used
G_3: Lockdown
G_2: Increased Awareness
G_4: Medical Supplies Available
G_1: Successful Media Campaigns
+
-

FIGURE 1. **A hypothetical goal model that policymakers can think to handle the coronavirus COVID-19 pandemic.**

Goal models are promising in the field of strategic planning and public policy decision support, where a goal model can serve as the depiction of policymaker understanding of how to achieve his final goals [19], [24]. Figure 1 illustrates an example goal model that policymakers can think to handle the coronavirus COVID-19 pandemic. In this figure, G_7 is the final, or *root* goal in the model, and all other goals contribute to each other and finally contribute to it. G_1, G_3, and G_4 are *leaf* goals that no other goals contribute to them. Goals may contribute positively or negatively to other goals. For example, G_6 contributes positively to G_8, while G_9 contributes back negatively to G_6.

A Policymaker may need to estimate which leaf goals need attention first (consequently, the required fund) for the final goal to be satisfied as early as possible. These estimates can be obtained from goal model evaluation techniques. Goal model evaluation techniques can be used to evaluate satisfaction levels of root goals from satisfaction levels of leaf goals. Some other techniques work top-down, trying to find an initial set of satisfaction levels of leaf goals that can lead to the satisfaction of root goals.

Qualitative approaches for goal model evaluation assign initial labels representing the satisfaction level to leaf goals [8]. These labels are then propagated upwards involving new labels to be assigned to the next goals directly, or in case of conflicts, requiring human interaction to resolve. In [11], the authors introduced a similar forward propagating algorithm adding the deniability of the goal as a second independent dimension to the labels assigned to goals, removing the chance of conflict, and the requirement for human interaction during the execution of the algorithm. Qualitative approaches for goal model evaluation yield coarse values that cannot be satisfactory when more fine-grained outputs are required. In [18], a quantitative approach for evaluating KAOS-based goal models was introduced based on a probabilistic approach to represent partial satisfaction of goals. The authors in [11] extended their qualitative approach and provided a quantitative version of their algorithm. Quantitative approaches for goal model evaluation based on fuzzy logic have been presented in [7], [27]. Further analysis of various goal model reasoning techniques is presented in [14].

Tools have been developed for the implementation of various satisfaction evaluation methodologies and algorithms. Among these tools is jUCMNav [26], which is a graphical editor and analysis tool for URN. It is open-source and based on the Eclipse platform. It supports the qualitative or quantitative evaluation of GRL models. jUCMNav has also been used in other research efforts as a platform for implementing other evaluation algorithms and adding specific features, e.g., [2], [22]. OpenOME is another open-source Eclipse-based graphical tool for modeling and analysis of requirements using $i*$ concepts. Another notable tool is CGM-Tool, which supports modeling and reasoning on constrained goal models, is written in Java, and is also based on Eclipse [20]. Most tools are dedicated to requirements engineering and have been developed in the context of specific methodology and model type. They require adaptations to be suitable for use in other techniques.

The mentioned satisfaction analysis techniques consider only what value the final goal will eventually take, ignoring the time profile of satisfaction levels of both leaf goals (the inputs) and the root goal (the output). Additionally, these techniques ignore the nature of specific goal satisfaction dynamics, e.g., the required time for goals to be satisfied after their precedents were fully satisfied.

This paper proposes a new methodology for goal model evaluation based on the state-space representation model that has extensive uses in control engineering. The proposed methodology allows using time series for the inputs to leaf goals, and obtain the time series of the satisfaction level of the final goal, which is the outcome from goal model evaluation. Specific goal dynamic characteristics (in terms of goal satisfaction latency) for specific goals can also be defined and considered, bringing more accurate and realistic results. This way, we can extend *what-if* scenarios into *what* and *when*, which can be useful in decision making.

A tool has also been implemented for graphical editing of goal models using *i∗* goal model conventions. The tool can also detect leaf goals allowing graphical manipulation of their satisfaction input levels and providing an instant visualization of the time profile of the satisfaction level of the output root goal. For the execution of the state-space model, the tool uses the signal module of SciPy [30], the free and open-source, cross-platform, widely available Python library for scientific and technical computing. This library was introduced around 2001 and is still actively maintained and used in various scientific and technical areas.

Section II presents a brief introduction to the state-space representation model of systems and its rationale. Section III describes the proposed methodology, the principles behind, and the notion of *goal model fusion*. The goal model fusion algorithm is then presented and is followed by a complete example to demonstrate the whole process and hence how the goal model can be evaluated. The results for that example are then presented and discussed in Section IV.

II. STATE-SPACE REPRESENTATION OF SYSTEMS

State-space representation of a system is a mathematical model of a physical system as a set of input, output, and state variables related by first-order differential equations. This model has been extensively used in modern control engineering. Today the approach is not only being used in the analysis and design of control systems but also became a well-known methodology in other areas like finance and economics. Moreover, state-space techniques gained great support for software tools, frameworks, and programming languages.

Comparing the state-space approach to classical control, which uses frequency domain, state-space can be used with time-variant, nonlinear systems of multiple inputs and multiple outputs [21].

For a system having output y_1 for an input u_1, and output y_2 for an input u_2, if the system has the output equals $y_1 + y_2$ for an input $u_1 + u_2$, then the system satisfies the property of superposition. Moreover, if a system has an output of y for an input u, and output of βy for an input βu, then the system is homogeneous. If a system follows superposition and is homogeneous, then the system is linear. Generally, the system can be defined by the two equations:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} \tag{1}$$

$$\vec{y} = C\vec{x} + D\vec{u} \tag{2}$$

Moreover, if system parameters, the matrices A, B, C, and D, do not change with time, the system is called time-invariant. Equation (1) specifies how the state can change with the time as a function of the current state value $\vec{x}$ and the system input $\vec{u}$ whereas Equation (2) relates $\vec{x}$ and $\vec{u}$ to the system output $\vec{y}$. Table 1 lists all components of equations (1) and (2) and summarizes the effect of the four system parameters on the system.

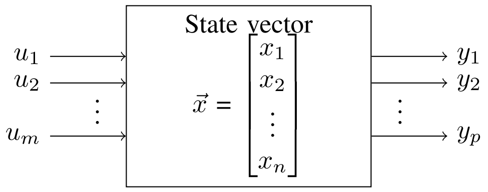

State vector
u_1
u_2
u_m
$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
y_1
y_2
y_p

FIGURE 2. State-space representation of a system.

TABLE 1. The components of Equations (1) and (2), and their characteristics.

Element	Dimension	Main Characteristics
$\vec{x}$	$n \times 1$	the state vector
$\dot{\vec{x}}$	$n \times 1$	the derivative of the state vector
$\vec{u}$	$m \times 1$	the input vector
$\vec{y}$	$p \times 1$	the output vector
A	$n \times n$	effect of $\vec{x}$ on $\dot{\vec{x}}$
B	$n \times m$	effect of $\vec{u}$ on $\dot{\vec{x}}$
C	$p \times n$	effect of $\vec{x}$ on $\vec{y}$
D	$p \times m$	direct effect of $\vec{u}$ on $\vec{y}$ (mostly a zero matrix)

III. THE PROPOSED FRAMEWORK

The proposed framework is based on the analogy between systems, which is a known technique that uses the solutions from one science field and applies it to another field of interest. To the best knowledge of the author, the analogy between control system theory and goal modeling has not been studied. A goal that contributes to another goal has some analogy to systems theory where the output from a system component can be introduced as input to another, forming a larger system.

In the state-space representation of a system, the system is viewed as a set of inputs, outputs, and state variables. The system is defined in terms of four matrices A, B, C, and D. These matrices relate these components to each other. The initial conditions of the system can also be defined as the value of the state vector at time 0. In the proposed

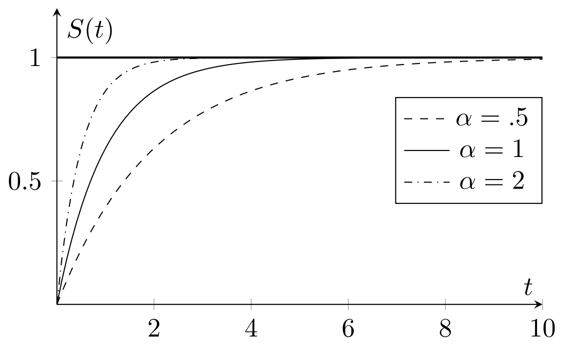


FIGURE 3. Achieving different goal satisfaction latencies based on α, in case of an input with the Heaviside step function.

methodology, goals are considered analogous to system components, so they can also be defined in terms of the four matrices A, B, C, and D as in Equations (1) and (2). For simplicity, we can omit the D matrix from Equation (2) as it always has a null value. In our proposed evaluation framework, a default set of matrices is initially assigned to each goal in the goal model. The objective is to get a single set of matrices with the same dynamic characteristics as the original goal model. This set of matrices directly relates the inputs to the leaf goals to the outputs of the root goals, which means that applying the same input values to both the original goal model, and to the reduced one would produce the same results at any time. The transformation of the original goal model into a single goal with a single set of matrices is called *goal model fusion*.

A goal is assigned the set of matrices initially: $A = [-1]$, $B = [1]$, $C = [1]$ representing the simple single-input single-output system defined by the equations $\dot{x} = u - x$ and $y = x$. If a goal has additional inputs, then the initial B matrix assigned to it should have a number of columns equal to the number of its inputs, with each column having a ratio representing the weight of participation of this particular input to all inputs. This simple single state system has a state vector $\vec{x} \in R^1$. The output of the system at any time is directly equal to this state value, and the next state variable value comes directly from the input u. From a goal model point of view, this represents a simple propagation latency from one goal to another. For example, if the input to a goal with this initial set of matrices, at time $t = 0$ is the Heaviside step function $u(t) = 1$, then the satisfaction of the goal will take the form of $S(t) = 1 - e^{-t}$. Due to the nature of individual goals in the goal models, a goal may require more time to reach satisfaction even if its input has already reached satisfaction. This variable latency can be expressed based on the parameter α in the equation $S(t) = 1 - e^{-\alpha t}$ as shown in Figure 3. Based on the techniques from control theory, the initial set of matrices to be assigned initially to a goal can then be $A = [-\alpha]$, $B = [\alpha]$, $C = [1]$.

A. GOAL MODEL FUSION

A goal model usually starts with a target goal, breaking it into several sub-goals. It then passes through a several breakdowns and refinement iterations until no further breakdown can be possible. From an evaluation point of view, the goal model can be viewed as a transformation that *indirectly* relates the external inputs that lead to leaf goals satisfactions to the satisfaction level of the root goal. Goal model fusion provides a means to relate the inputs of the leaf goals *directly* to the achievement of root goals.

Goal model fusion is an iterative process where every step involves the fusion of the furthest goal into its next goals. The furthest goal is the one having the maximum number of hops in its shortest path to the root goal. Each step combines matrices, states, and inputs of the further goal into its subsequent goals without losing any information. The fusion process ends with a single goal representing the whole goal model with a single set of matrices maintaining all the original model dynamics.

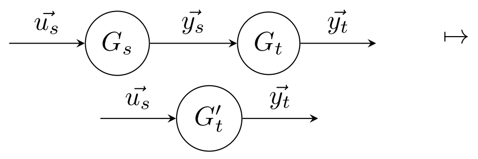


FIGURE 4. Fusion of G_s into its next goal G_t.

For a simple configuration, as seen in Figure 4, let G_s be the furthest goal from the root goal, which should then be fused into G_t. G_t gets input from G_s only, so its B matrix has a single column $B = [1]$, and $u_t = y_s$. G_t is said to be extended by G_s. Starting with Equations (1) and (2):

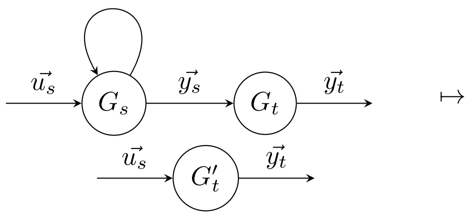
$$\begin{aligned}\dot{\vec{x_t}} &= A_t\vec{x_t} + B_t\vec{u_t}\\ &= A_t\vec{x_t} + B_t\vec{y_s}\\ &= A_t\vec{x_t} + B_tC_s\vec{x_s},\\ \dot{\vec{x_s}} &= A_s\vec{x_s} + B_s\vec{u_s},\\ \vec{y_t} &= C_t\vec{x_t}\end{aligned}$$

which can be written in matrix form as:

$$\begin{bmatrix}\dot{\vec{x_t}}\\ \dot{\vec{x_s}}\end{bmatrix} = \begin{bmatrix}A_t & B_tC_s\\ 0 & A_s\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix} + \begin{bmatrix}0\\ B_s\end{bmatrix}\vec{u_s}$$

$$\vec{y_t} = \begin{bmatrix}C_t & 0\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix}$$

These two equations suggest a new goal G'_t as a replacement of the original G_t after fusion of G_s along with its inputs $\vec{u_s}$ into it. The new equivalent goal G'_t has $A'_t = \begin{bmatrix}A_t & B_tC_s\\ 0 & A_s\end{bmatrix}$, $B'_t = \begin{bmatrix}0\\ B_s\end{bmatrix}$, $C'_t = \begin{bmatrix}C_t & 0\end{bmatrix}$. If the goal model has loops, then the inherited input from G_s to G'_t, $\vec{u_s}$, may also include output from other goals, in addition to external inputs that come from outside the goal model to leaf goals. In the case of G_s affecting more than one target (e.g., G_2 contribution to G_5 and G_6 in Figure 6(c)), then the same process is applied to each of these targets before G_s can be removed from the goal model.



$$\begin{tikzcd} \arrow[r, "\vec{u_s}"] & G_s \arrow[loop above] \arrow[r, "\vec{y_s}"] & G_t \arrow[r, "\vec{y_t}"] & {} \end{tikzcd} \mapsto \begin{tikzcd} \arrow[r, "\vec{u_s}"] & G_t' \arrow[r, "\vec{y_t}"] & {} \end{tikzcd}$$

FIGURE 5. Fusion of a goal with a self-loop into its next goal.

The next goal G_t may possibly have inputs from goals other than G_s. In this case, B_t is divided horizontally into m sections corresponding to the number of the inputs of G_t, with the input from G_s, y_s one of them.

$$\begin{aligned}\dot{\vec{x_t}} &= A_t\vec{x_t} + \begin{bmatrix}B_{t_1} & B_{t_2} & \dots & B_{t_m}\end{bmatrix}\begin{bmatrix}\vec{u_{t_1}}\\ \vec{u_{t_2}}\\ \vdots\\ \vec{u_{t_m}}\end{bmatrix}\\ &= A_t\vec{x_t} + \begin{bmatrix}B_{t_1} & B_{t_2} & \dots & B_{t_m}\end{bmatrix}\begin{bmatrix}\vec{y_s}\\ \vec{u_{t_2}}\\ \vdots\\ \vec{u_{t_m}}\end{bmatrix}\\ &= A_t\vec{x_t} + B_{t_1}C_s\vec{x_s} + \begin{bmatrix}0 & B_{t_2} & \dots & B_{t_m}\end{bmatrix}\begin{bmatrix}0\\ \vec{u_{t_2}}\\ \vdots\\ \vec{u_{t_m}}\end{bmatrix},\\ \dot{\vec{x_s}} &= A_s\vec{x_s} + B_s\vec{u_s},\\ \vec{y_t} &= C_t\vec{x_t}\end{aligned}$$

which can be written as:

$$\begin{bmatrix}\dot{\vec{x_t}}\\ \dot{\vec{x_s}}\end{bmatrix} = \begin{bmatrix}A_t & B_{t_1}C_s\\ 0 & A_s\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix} + \begin{bmatrix}0 & B_{t_2} & \dots & B_{t_m}\\ B_s & 0 & \dots & 0\end{bmatrix}\begin{bmatrix}\vec{u_s}\\ \vec{u_{t_2}}\\ \vdots\\ \vec{u_{t_m}}\end{bmatrix}$$

$$\vec{y_t} = \begin{bmatrix}C_t & 0\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix}$$

suggesting a new goal G_t' after fusion of G_s along with its inputs $\vec{u_s}$ into G_t with $A_t' = \begin{bmatrix}A_t & B_{t_1}C_s\\ 0 & A_s\end{bmatrix}$, $B_t' = \begin{bmatrix}0 & B_{t_2} & \dots & B_{t_m}\\ B_s & 0 & \dots & 0\end{bmatrix}$, and $C_t' = \begin{bmatrix}C_t & 0\end{bmatrix}$

In goal models with loops, the fusion of leaf goals into their next goals eventually results in a goal with a self-loop like the one shown in Figure 5

$$\begin{aligned}\dot{\vec{x_t}} &= A_t\vec{x_t} + B_t\vec{y_s}\\ &= A_t\vec{x_t} + B_tC_s\vec{x_s},\\ \dot{\vec{x_s}} &= A_s\vec{x_s} + \begin{bmatrix}B_{s_1} & B_{s_2}\end{bmatrix}\begin{bmatrix}\vec{u_s}\\ \vec{y_s}\end{bmatrix}\\ &= A_s\vec{x_s} + \begin{bmatrix}B_{s_1} & 0\end{bmatrix}\begin{bmatrix}\vec{u_s}\\ 0\end{bmatrix} + B_{s_2}C_s\vec{x_s},\\ \vec{y_t} &= C_t\vec{x_t}\end{aligned}$$

In matrix form, the above equations can be written as:

$$\begin{bmatrix}\dot{\vec{x_t}}\\ \dot{\vec{x_s}}\end{bmatrix} = \begin{bmatrix}A_t & B_tC_s\\ 0 & A_s + B_{s_2}C_s\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix} + \begin{bmatrix}0\\ B_{s_1}\end{bmatrix}\vec{u_s}$$

$$\vec{y_t} = \begin{bmatrix}C_t & 0\end{bmatrix}\begin{bmatrix}\vec{x_t}\\ \vec{x_s}\end{bmatrix}$$

forming a new goal G_t' with input u_s only, $A_t' = \begin{bmatrix}A_t & B_tC_s\\ 0 & A_s + B_{s_2}C_s\end{bmatrix}$, $B_t' = \begin{bmatrix}0\\ B_{s_1}\end{bmatrix}$, and $C_t' = \begin{bmatrix}C_t & 0\end{bmatrix}$

B. FUSION OF GOAL MODELS SUPPORTING NEGATIVE CONTRIBUTION

So far, a goal is associated with a value that represents its satisfaction level ranging from 0, which represents total denial, to 1, which represents total satisfaction. This is guaranteed by starting with inputs that fall in this range. For goals with multiple inputs, weights are used to fulfill this property. To support goal models with possible negative contributions between goals, e.g., the relation between G_9 and G_6 in the goal model in Figure 1, we need to keep the goals' denial values in a separate channel. Each connection from one goal to another in the goal model is represented by two separate channels; the suffix s will be used to denote the satisfaction channel, and the suffix d for the denial channel. The values in each channel still range from 0 to represent no satisfaction/denial to 1 for complete satisfaction/denial. The absolute satisfaction/denial level of a goal can then be determined by subtracting the denial value from the satisfaction value of the goal, resulting in a value ranging from -1 for total denial to 1 for total satisfaction, which can be normalized again to a value that ranges from 0 to 1.

For positive contribution relation from a goal G_s to another goal G_t, the s-channel of G_s is connected to the s-channel of G_t and the same is for the d-channel. The satisfaction of G_s leads to satisfaction in G_t and denial of G_s also leads to denial of G_t. On the other hand, for negative contribution, the s-channel of G_s is connected to the d-channel of G_t, and the d-channel of G_s is connected to the s-channel of G_t. This exchange in channels reflects that the satisfaction of G_s leads to a denial of G_t and vice versa.

If a goal has connections from m other goals, then the actual input vector $\vec{u}$ has $2m$ components since each connection from these goals requires two inputs, one for the satisfaction channel, and the other for the denial channel, i.e., $\vec{u} = \begin{bmatrix}u_{1_s} & u_{1_d} & u_{2_s} & u_{2_d} & \dots & u_{m_s} & u_{m_d}\end{bmatrix}^T$. The goal output $\vec{y}$ now has always two outputs, again one for the s-channel and the other is for the d-channel, i.e., $\vec{y} = \begin{bmatrix}y_s\\ y_d\end{bmatrix}$. For a goal that has n states, A, B, and C matrices have the dimensions of $n \times n$, $n \times 2m$, and $2 \times n$ respectively:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix},$$

$$B = \begin{bmatrix} b_{11_s} & b_{11_d} & b_{12_s} & b_{12_d} & \dots & b_{1m_s} & b_{1m_d} \\ b_{21_s} & b_{21_d} & b_{22_s} & b_{22_d} & \dots & b_{2m_s} & b_{2m_d} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n1_s} & b_{n1_d} & b_{n2_s} & b_{n2_d} & \dots & b_{nm_s} & b_{nm_d} \end{bmatrix},$$

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \end{bmatrix}$$

According to the latency parameter α of a goal, the initial value of A matrix will be $\begin{bmatrix} -\alpha & 0 \\ 0 & -\alpha \end{bmatrix}$, and C will take $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. For a goal with a single input, B will be assigned the value $\begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}$. For a goal with m-inputs, B will take the form $\begin{bmatrix} B_1 & B_2 & \dots & B_m \end{bmatrix}$, where each part of this matrix B_i corresponds and input i with weight w_i. For a positive contributing input i, $B_i = w_i \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}$, and for negative contribution, $B_i = w_i \begin{bmatrix} 0 & \alpha \\ \alpha & 0 \end{bmatrix}$. The sum of all w_i should always equal to 1.

C. GOAL MODEL FUSION ALGORITHM

The goal model is defined in terms of a set of goals G, and a set of links between them L. The objective of the algorithm is to simplify the goal model into a single goal and a set of inputs. The algorithm achieves this by finding the furthest leaf goal and merging it into the goals it contributes to, maintaining the inputs of the leaf goal. This operation is repeated until no further fusion is possible.

Algorithm 1 Goal Model Fusion G, L

```
1:  procedure GoalModelFusion($G, L$)
2:      $G_{root}$ ← FindRootGoal($G, L$)
3:      while Length($L$) > 0 do    ▷ Links between goal still exist
4:          $G_1$ ← FurthestGoal($G_{root}, G, L$)
5:          $Targets$ ← OutgoingFrom($G_1, G, L$)
6:          for all $G_2 \in Targets$ do
7:              $G_2$ ← Combine($G_1, G_2$)
8:          end for
9:          DeleteGoal($G_1$)
10:     end while
11: end procedure
```

D. EXAMPLE

The goal model in Figure 6(a) corresponds to the example goal model in Figure 1. u_1, u_3, and u_4 correspond to the

```
1:  procedure FurthestGoal($G_{root}, G, L$)
2:      $MAX\_DEPTH$ ← 999999
3:      $Depths$ ← InitializeList($G.length, MAX\_DEPTH$)
4:      $Depths[G_{root}]$ ← 0
5:      while $MAX\_DEPTH \in Depths$ do
6:          for all $l \in L$ do
7:              if $Depths[l.to] \neq MAX\_DEPTH$ then
8:                  $Depths[l.from]$ ← Min($Depths[l.from], Depths[l.to] + 1$)
9:              end if
10:         end for
11:     end while
12:     $G_{furthest}$ ← $G_{root}$
13:     for all $g \in G$ do
14:         if $Depths[g] > Depths[G_{root}]$ then
15:             $G_{furthest}$ ← $g$
16:         end if
17:     end for
18:     return $G_{furthest}$
19: end procedure
```

```
1:  procedure Combine($G_s, G_t$)
2:      $n_t$ ← SIZE($A_t$)
3:      $n_s$ ← SIZE($A_s$)
4:      $n$ ← $A_t + A_s$
5:      $A$ ← ZeroMatrix($n, n$)
6:      SubMat($A, 0 : n_t, 0 : At.width$) ← $A_t$
7:      SubMat($A, n_t : n, 0 : As.width$) ← $A_s$
8:      $i$ ← the index corresponding to $G_s$ in $G_t.inputs$
9:      $B_i$ ← the part of $B_t$ corresponding to $i$
10:     SubMat($A, 0 : n_t, n_t : n$) ← $B_i \times C_s$
11:     if $G_s.inputs$ contains input from $G_s$ then    ▷ A self-loop in $G_s$
12:         $j$ ← the index corresponding to $G_s$ in $G_s.inputs$
13:         $B_j$ ← the part of $B_s$ corresponding to $j$
14:         SubMat($A, n_t : n, n_t : n$) ← SubMat($A, n_t : n, n_t : n$) $+ B_j \times C_s$
15:         Delete the $j^{th}$ part of $B_s$ and $B_s.inputs$
16:     end if
17:     $m$ ← Length($G_t.inputs$) + Length($G_s.inputs$) − 1
18:     $B$ ← ZeroMatrix($n, m$)
19:     SubMat($B, 0 : n_t, 0 : i$) ← SubMat($B_t, :, 0 : i$)
20:     SubMat($B, n_t : n, i : i + B_s.width$) ← $B_s$
21:     SubMat($B, 0 : n_t, i + B_s.width : m$) ← SubMat($B_t, :, i + 1 : B_t.width$)
22:     $C$ ← ZeroMatrix($1, n$)
23:     SubMat($C, :, 0 : 1]$) ← Identity(1)
24:     $inputs$ ← SubList($G_t.inputs, 0 : i$)
25:     $inputs$ ← ListExtend($inputs, G_s.inputs$)
26:     $l$ ← SubList($G_t.inputs, i + 1 :$)
27:     $inputs$ ← ListExtend($inputs, l$)
28:     DeleteRelation($G_s, G_t$)
29:     return Goal($A, B, C, inputs$)
30: end procedure
```

external inputs that lead to the satisfaction of the leaf goals G_1, G_3, and G_4. All goals are assigned A matrix with a value of $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$, and C matrix with a value of $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Goals G_1, G_2, G_3, G_4, G_7, and G_9 have a single input, so they are assigned B matrix with a value of $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, whereas G_5 and G_8 have two inputs each, so they are assigned B matrix with a value of $\begin{bmatrix} .5 & 0 & .5 & 0 \\ 0 & .5 & 0 & .5 \end{bmatrix}$. B_6 has a value of $\begin{bmatrix} .33 & 0 & .33 & 0 & 0 & .33 \\ 0 & .33 & 0 & .33 & .33 & 0 \end{bmatrix}$ corresponding to the positive contribution from G_3 and G_2, and the negative contribution from G_9. Negative contribution results in a vertical flipping of its corresponding part of matrix B.

Round 1: The furthest goal from the root goal G_7 is G_1. G_1 is combined into its target G_2, leaving

$$A_2 = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

u_1 (which is inherited from G_1) replaces y_1 in G_2 inputs. The relation from G_1 to G_2 is deleted, and G_1 is then removed from the model.

Round 2: G_2, G_3, G_4, G_9 now have the same distance from G_7, so any of them can be selected. We arbitrarily take G_2. G_2 has contributions to both G_6 and G_5, resulting in

$$A_6 = \begin{bmatrix} -1 & 0 & .33 & 0 & 0 & 0 \\ 0 & -1 & 0 & .33 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_6 = \begin{bmatrix} 0 & 0 & .33 & 0 & 0 & .33 \\ 0 & 0 & 0 & .33 & .33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$C_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

u_1 replaces y_2 in G_6 inputs, and the relation from G_2 to G_6 is deleted. For G_5,

$$A_5 = \begin{bmatrix} -1 & 0 & .5 & 0 & 0 & 0 \\ 0 & -1 & 0 & .5 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_5 = \begin{bmatrix} .5 & 0 & 0 & 0 \\ 0 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad C_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

u_1 replaces y_2 in G_5 inputs, and the relation from G_2 to G_5 is deleted. G_2 can then be removed from the model.

Round 3: G_3 is combined into G_6 leaving

$$A_6 = \begin{bmatrix} -1 & 0 & .33 & 0 & 0 & 0 & .33 & 0 \\ 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & .33 \\ 0 & 0 & 0 & 0 & .33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$C_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

u_3 replaces y_3 in G_6 inputs, and the relation from G_3 to G_6 is deleted, and G_3 can then be removed.

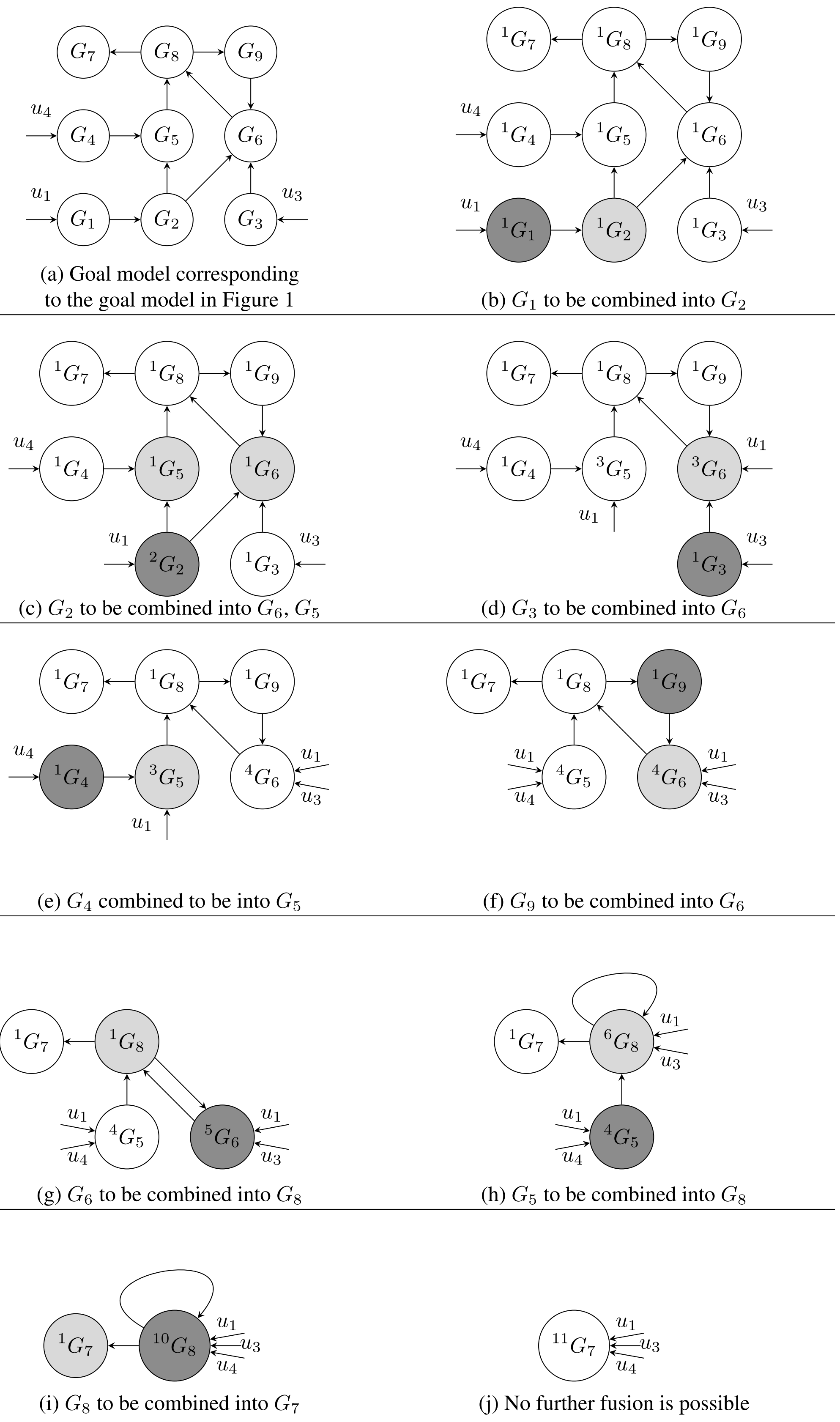
Round 4: G_4 is the furthest goal to G_7; it is then combined into its target G_5.

$$A_5 = \begin{bmatrix} -1 & 0 & .5 & 0 & 0 & 0 & .5 & 0 \\ 0 & -1 & 0 & .5 & 0 & 0 & 0 & .5 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$C_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

u_4 replaces y_4 in G_5 inputs, and the relation from G_4 to G_5 is deleted, and G_4 can then be removed.



(a) Goal model corresponding to the goal model in Figure 1

(b) G_1 to be combined into G_2

(c) G_2 to be combined into G_6, G_5

(d) G_3 to be combined into G_6

(e) G_4 combined to be into G_5

(f) G_9 to be combined into G_6

(g) G_6 to be combined into G_8

(h) G_5 to be combined into G_8

(i) G_8 to be combined into G_7

(j) No further fusion is possible

FIGURE 6. Fusion of the example goal model in Figure 1.

Round 5: G_9 is combined into its target G_6.

$$A_6 = \begin{bmatrix} -1 & 0 & .33 & 0 & 0 & 0 & .33 & 0 & 0 & .33 \\ 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & .33 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$C_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

y_8 replaces y_9 in G_6 inputs, and the relation from G_9 to G_6 is deleted, along with G_9.

Round 6: G_6 is combined into its target G_8.

$$A_8 = \begin{bmatrix} -1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & 0 & 0 & .33 \\ 0 & 0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & .33 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$B_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$C_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

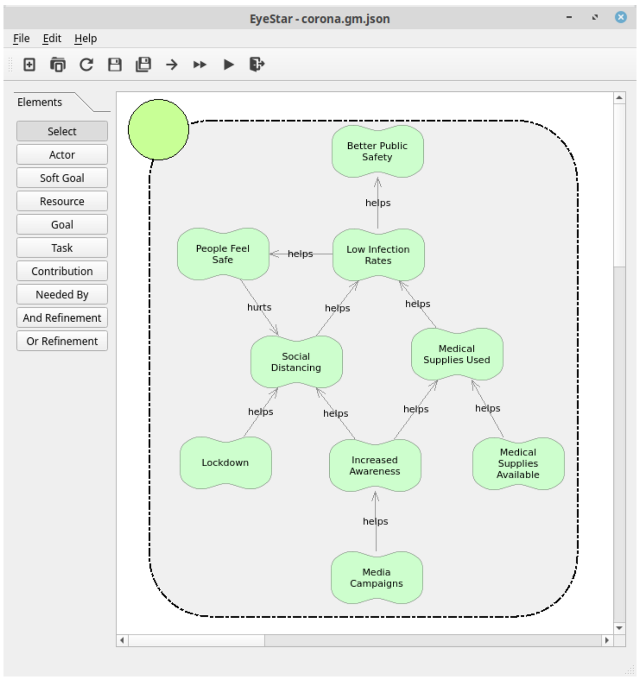


FIGURE 7. The main window of the goal model editing tool.

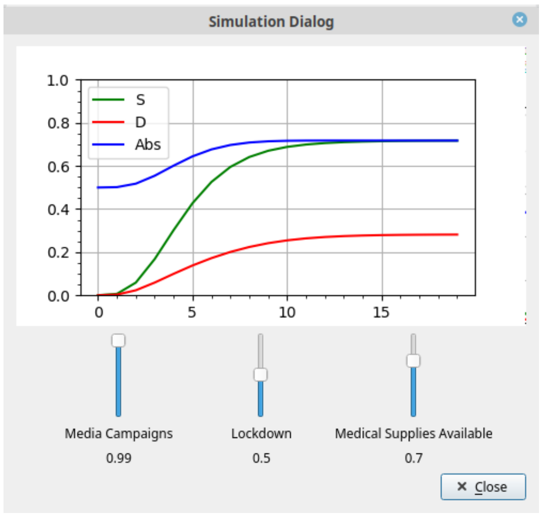


FIGURE 8. Goal model evaluation dialog.

u_1, u_3, and y_8 replace y_6 in G_8 inputs, and the relation from G_6 to G_8 is deleted, along with G_6.

Round 7: G_5 is combined into its target G_8, shown at the bottom of the next page.

Another instance of u_1 and u_4 replace y_5 in G_8 inputs, and the relation from G_5 to G_8 is deleted, along with G_5.

Round 8: G_8 is combined into the root goal G_7, shown at the bottom of the 11th page..

The two instances of u_1 along with u_3 and u_4 replace y_8 in G_7 inputs resolving the loop in G_8. The relation from G_8 to G_7 is deleted as well as G_8. Now, the goal model has been wholly fused into its root goal G_7, and the matrices A_7, B_7, C_7 fully determine the dynamic behavior of the original goal model. Using traditional control engineering tools can then be used to evaluate and analyze the effect of the original inputs u_1, u_3, and u_4 on the satisfaction level of the root goal G_7.

IV. RESULTS

A tool has been developed to implement the proposed methodology for goal model evaluation. The tool includes a graphical user interface (based on PyQT5) for editing goal models based on $i*$ concepts and terminologies. A screenshot of the main window of the tool is shown in Figure 7. Upon triggering the evaluation feature, the goal model fusion algorithm is executed, yielding a set of three matrices defining the relation between the external inputs and the root goal. A window is then shown featuring a set of slider widgets corresponding to the external inputs of the goal model (u_1: input to "G_1: Successful Media Campaigns", u_3: input to "G_3: Lockdown", u_4: input to "G_4: Medical Sup-plies Available"). Figure 8 is a screenshot of that window, showing the curves of satisfaction level (S), the denial level (D), and the absolute satisfaction level (Abs) of the root goal "G_7: Better Public Safety". These curves are instantly updated

$$A_8 = \begin{bmatrix}
-1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & 0 & 0 & .33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & .33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & .5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & .5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix},$$

$$B_8 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix},$$

$$C_8 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.$$

upon changing the values of the external inputs using the slider widgets.

The framework is tested with satisfaction levels 0, 0.5, and 1 for each of the external inputs of the goal model. The obtained results are listed in Table 2. The input values u_1, u_3, and u_4 in the table are introduced as a Heaviside step function time series of a length of 20. The output at $t = 5$ and at $t = 20$ (which can be considered the steady-state in this example) is sampled for each input in the table. These results are based on a fixed value of 1 for the latency factor α for all goals, and a fixed weight value of $\frac{1}{N}$ for N inputs contributing to the same goal.

The output at $t = 0$ for all input values is not listed since it always has the results $S = 0, D = 0$ due to the delay required for goals to propagate their values. The steady-state output for a complete denial of all external inputs (0, 0, 0) is $S = 0.142, D = 0.857$. This nonzero satisfaction and incomplete denial come from the negative contribution from G_9 to G_6, which is the reason also for a complete satisfaction input (1, 1, 1) yielding $S = 0.857, D = 0.142$, incomplete satisfaction and nonzero denial. Complete satisfaction in u_1 with a complete denial in other inputs leads to a $S = 0.499, D = 0.499$, whereas a complete satisfaction in either u_3 or u_4 with a complete denial in other inputs yields lower absolute satisfaction levels. These results reflect the fact that G_2, which is the next goal of G_1, has a double effect, contributing to two goals simultaneously (G_5 and G_6). The reason for u_3 is less significant than u_4 is that G_3 affects G_6, which has negative

$$A_7 = \begin{bmatrix}
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & 0 & 0 & .33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -1 & 0 & .33 & 0 & 0 & 0 & .33 & .33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & .5 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & .5 & 0 & 0 & 0 & .5\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix},$$

$$B_7 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix},$$

$$C_7 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.$$

TABLE 2. Evaluation of the goal model in Figure 1 against external inputs.

u_1	u_3	u_4	$S_{t=5}$	$D_{t=5}$	$S_{s.s.}$	$D_{s.s.}$
0.000	0.000	0.000	0.115	0.828	0.142	0.857
0.000	0.000	0.500	0.223	0.721	0.249	0.749
0.000	0.000	1.000	0.330	0.613	0.357	0.642
0.000	0.500	0.000	0.187	0.756	0.214	0.785
0.000	0.500	0.500	0.294	0.649	0.321	0.678
0.000	0.500	1.000	0.401	0.542	0.428	0.571
0.000	1.000	0.000	0.258	0.685	0.285	0.714
0.000	1.000	0.500	0.366	0.577	0.392	0.607
0.000	1.000	1.000	0.473	0.470	0.499	0.499
0.500	0.000	0.000	0.293	0.651	0.321	0.678
0.500	0.000	0.500	0.400	0.543	0.428	0.571
0.500	0.000	1.000	0.507	0.436	0.535	0.464
0.500	0.500	0.000	0.364	0.579	0.392	0.607
0.500	0.500	0.500	0.472	0.472	0.499	0.499
0.500	0.500	1.000	0.579	0.364	0.607	0.392
0.500	1.000	0.000	0.436	0.507	0.464	0.535
0.500	1.000	0.500	0.543	0.400	0.571	0.428
0.500	1.000	1.000	0.651	0.293	0.678	0.321
1.000	0.000	0.000	0.470	0.473	0.499	0.499
1.000	0.000	0.500	0.577	0.366	0.607	0.392
1.000	0.000	1.000	0.685	0.258	0.714	0.285
1.000	0.500	0.000	0.542	0.401	0.571	0.428
1.000	0.500	0.500	0.649	0.294	0.678	0.321
1.000	0.500	1.000	0.756	0.187	0.785	0.214
1.000	1.000	0.000	0.613	0.330	0.642	0.357
1.000	1.000	0.500	0.721	0.223	0.749	0.249
1.000	1.000	1.000	0.828	0.115	0.857	0.142

feedback from G_9, while the path starting from G_4 has not.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced a framework for evaluating goal models based on control system theories. The proposed methodology considers the time-varying inputs resulting in a time-varying output. It also considers specific goal dynamic characteristics. An algorithm has been introduced for deducing the state-space matrices for the whole goal model from individual goals. Representing a goal model in terms of a few matrices would hypothetically give better evaluation computation performance than other evaluation techniques. Improving the computation time of the evaluation process allows for more combinations of inputs to be tested against the system being modeled. Furthermore, this new view of the goal model opens a new area for further analysis of the goal model dynamics.

A supporting tool has been developed to prove the concept of this new direction. This tool has been used to obtain a state-space model with the same characteristics as the original goal model. Execution of the state-space model against different input values is done using a mature, widely available, actively maintained Python package for technical computing.

A limitation of the proposed framework is that it cannot support nonlinear relations among goals, such as having alternatives for goal satisfaction. Future work may include a linearization for such relations. According to the new view of a goal model as a system of systems, further analysis of the goal model dynamics can be achieved. Parameters like initial matrices, α for each goal, inputs weights can be time-varying for more realistic results. Using the control theory concepts, it might also be possible to find an initial set of inputs that can lead to particular satisfaction of root goals at some point in time.

REFERENCES

[1] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, nos. 1–2, pp. 3–50, Apr. 1993.

[2] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the goal-oriented requirement language," *Int. J. Intell. Syst.*, vol. 25, no. 8, pp. 841–877, Jun. 2010.

[3] A. I. Anton, "Goal-based requirements analysis," in *Proc. 2nd Int. Conf. Requirements Eng.*, Apr. 1996, pp. 136–144.

[4] A. I. Antón, W. M. McCracken, and C. Potts, "Goal decomposition and scenario analysis in business process reengineering," in *Advanced Information Systems Engineering*, G. Wijers, S. Brinkkemper, and T. Wasserman, Eds. Berlin, Germany: Springer, 1994, pp. 94–104.

[5] Y. Asnar, P. Giorgini, and N. Zannone, "Reasoning about risk in agent's deliberation process: A Jadex implementation," in *Proc. 8th Int. Workshop Agent-Oriented Softw. Eng. (AOSE)*, Honolulu, HI, USA. Berlin, Germany: Springer, 2008, pp. 118–131, doi: 10.1007/978-3-540-79488-2_9.

[6] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Auto. Agents Multi-Agent Syst.*, vol. 8, no. 3, pp. 203–236, May 2004.

[7] G. Chatzikonstantinou and K. Kontogiannis, "Efficient parallel reasoning on fuzzy goal models for run time requirements verification," *Softw. Syst. Model.*, vol. 17, no. 4, pp. 1339–1364, Oct. 2018, doi: 10.1007/s10270-016-0562-9.

[8] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering* (International Series in Software Engineering), vol. 5. New York, NY, USA: Springer, Oct. 2000, doi: 10.1007/978-1-4615-5269-7.

[9] F. Dalpiaz, X. Franch, and J. Horkoff, "IStar 2.0 language guide," 2016, *arXiv:1605.07767*. [Online]. Available: http://arxiv.org/abs/1605.07767

[10] X. Franch, L. López, C. Cares, and D. Colomer, "The i* framework for goal-oriented modeling," in *Domain-Specific Conceptual Modeling*. Springer, 2016, pp. 485–506.

[11] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," in *Conceptual Modeling—ER*, S. Spaccapietra, S. T. March, and Y. Kambayashi, Eds. Berlin, Germany: Springer, 2003, pp. 167–181.

[12] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in *Proc. IEEE 19th Int. Requirements Eng. Conf.*, Aug. 2011, pp. 79–88.

[13] J. Horkoff and E. Yu, "Analyzing goal models: different approaches and how to choose among them," in *Proc. 2011 ACM Symp. Appl. Comput.*, 2011, pp. 675–682.

[14] J. Horkoff and E. Yu, "Comparison and evaluation of goal-oriented satisfaction analysis techniques," *Requirements Eng.*, vol. 18, no. 3, pp. 199–222, Sep. 2013.

[15] *User Requirements Notation (URN)–Language Definition*, document Z ITU-T. 151, Nov. 2008.

[16] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proc. 18th IEEE Int. Requirements Eng. Conf.*, Sep./Oct. 2010, pp. 115–124.

[17] A. Lapouchnian, *Goal-oriented requirements engineering: An overview of the Current Research*, vol. 32. Toronto, ON, Canada: Univ. Toronto, 2005.

[18] E. Letier and A. van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," *ACM SIGSOFT Softw. Eng. Notes*, vol. 29, no. 6, pp. 53–62, Nov. 2004, doi: 10.1145/1041685.1029905.

[19] Q. Ma and S. de Kinderen, "Goal-based decision making," in *Proc. Int. Work. Conf. Requirements Eng., Found. Softw. Qual.* Springer, 2016, pp. 19–35.

[20] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Multi-objective reasoning with constrained goal models," *Requirements Eng.*, vol. 23, no. 2, pp. 189–225, Jun. 2018.
[21] K. Ogata, *Modern Control Engineering*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2019.
[22] A. Pourshahid, D. Amyot, P. Chen, M. Weiss, and A. J. Forster, "Business process monitoring and alignment: An approach based on the user requirements notation and business intelligence tools," in *Proc. WER*, 2007, pp. 80–91.
[23] A. Pourshahid, D. Amyot, L. Peyton, S. Ghanavati, P. Chen, M. Weiss, and A. J. Forster, "Business process management with the user requirements notation," *Electron. Commerce Res.*, vol. 9, no. 4, pp. 269–316, Dec. 2009.
[24] A. Pourshahid, I. Johari, G. Richards, D. Amyot, and O. S. Akhigbe, "A goal-oriented, business intelligence-supported decision-making methodology," *Decis. Anal.*, vol. 1, no. 1, p. 9, Dec. 2014.
[25] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *Proc. ICMAS*, vol. 95, 1995, pp. 312–319.
[26] J.-F. Roy, J. Kealey, and D. Amyot, "Towards integrated tool support for the user requirements notation," in *Proc. Int. Workshop Syst. Anal. Modeling*. Springer, 2006, pp. 198–215.
[27] C. M. Subramanian, A. Krishna, A. Kaur, and R. P. Gopalan, "Quantitative reasoning of goal satisfaction in the i* framework," in *Proc. 27th Int. Conf. Softw. Eng. Knowl. Eng.* Pittsburgh, PA, USA: KSI Research Inc. and Knowledge Systems Institute Graduate School, Jul. 2015, pp. 666–669, doi: 10.18293/SEKE2015-158.
[28] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proc. 5th IEEE Int. Symp. Requirements Eng.*, Feb. 2001, pp. 249–262.
[29] A. van Lamsweerde, "Reasoning about alternative requirements options," in *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 380–397.
[30] P. Virtanen, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Feb. 2020.
[31] E. Yu, "Modelling strategic relationships for process reengineering," Ph.D. dissertation, Graduate Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1995.
[32] E. S. K. Yu and J. Mylopoulus, "Using goals, rules and methods to support reasoning in business process reengineering," *Intell. Syst. Accounting, Finance Manage.*, vol. 5, no. 1, pp. 1–13, Mar. 1996.
[33] I. El-Maddah and T. Maibaum, "Goal-oriented requirements analysis for process control systems design," in *Proc. 1st ACM IEEE Int. Conf. Formal Methods Models Co-Design*, 2003, pp. 45–46.

MOHAMED ABDEL-MONEM was born in Egypt, in 1975. He received the M.Sc. degree in electrical and computer engineering from the Military Technical College, Cairo, Egypt, in 2003. He is currently pursuing the Ph.D. degree with the Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University. He started his career as a Software Engineer, architecting and implementing ICT systems for various industries using various technologies and programming languages. Since he has retired from the military, he became a Freelance Software Engineering Consultant for several parties, including some governmental authorities. He also became an Entrepreneur using open-source software development technologies, which he has always been a constant supporter of them. His current research interests include conceptual modeling and software engineering.

ISLAM EL-MADDAH received the Ph.D. degree from the University of London, U.K., in 2004. He was working as a Professor Assistant with Taibah University, Saudi Arabia, from 2009 to 2011. He is currently working with the Faculty of Engineering, Ain Shams University, Egypt, as an Assistant Professor. He is interested in software engineering development and research, especially requirements analysis using goal models as well as software visualization. He is developing courses related to software engineering, including software formal specification, advanced software engineering, software testing, verification and validation, program analysis, and visualization.

HANI KAMAL MAHDI received the Ph.D. degree from Technische Universität Braunschweig, Germany, in 1984. He was a Postdoctoral Research Fellow with the Electrical and Computer Engineering Department, The Pennsylvania State University, PA, USA, from 1988 to 1989. He was a Visiting Professor with the Computer Vision and Image Processing (CVIP) Laboratory, Electrical Engineering Department, University of Louisville, KY, USA, from 2001 to 2002. He was on leave from Al-Isra University, Amman, Jordan; El-Emarat University, Al Ain, United Arab Emirates; and the Technology College, Hufuf, Saudi Arabia. He is currently a Professor of Computer Systems with the Faculty of Engineering, Ain Shams University, Cairo, Egypt. He was a recipient of the Distinguished University Award of Ain Shams University, in 2015. He received the Best (First) Graduated Students from the Electrical Engineering Department, Faculty of Engineering, Communication and Electronics Section, Ain Shams University, in 1971.