
Proceedings Paper:
Calinescu, Radu and Nunes Rodrigues, Genaína (2023) Goal Controller Synthesis for Self-Adaptive Systems. In: FormaliSE International Conference on Formal Methods in Software Engineering. FME Workshop on Formal Methods in Software Engineering. IEEE.

https://doi.org/10.1109/FormaliSE58978.2023.00008

Goal Controller Synthesis for Self-Adaptive Systems

Radu Calinescu
Department of Computer Science
University of York
York, UK
radu.calinescu@york.ac.uk

Genaína Nunes Rodrigues
Department of Computer Science
University of Brasília
Brasília, Brazil
genaina@unb.br

To change your mind is the best evidence you have one.

Desmond Ford

***Abstract*—Much like humans, a growing number of software-controlled systems must cope with uncertainty and disruption through self-adaptation. However, while humans achieve this feat by changing both the means through which they pursue their goals and—when unavoidable—the goals themselves, *self-adaptive systems* are often only using the former adaptation mechanism. In this 'research ideas' paper, we argue that exploiting the latter mechanism is equally important, and we propose a new goal modelling paradigm that supports reasoning about goal change, and the use of probabilistic model checking tools to synthesise goal-management control software for self-adaptive systems.**

I. Introduction

Software-controlled systems ranging from hospital cleaning robots [1] to infrastructure inspection drones [2] have the potential to help humans with activities that are tedious, tiring or dangerous. To achieve this potential, these systems need to mitigate the uncertainty and disruption encountered in their environments through *self-adaptation* [3], [4].

Two main adaptation mechanisms are available for self-adaptive systems (SAS). First, they can change their configuration. For instance, a robot may change the speed or force used to perform a task, or the probability of retrying the task after a failure. Second, they can modify their goals. As an example, a robot unable to deep clean a hospital room due to obstacles or time constraints may instead perform a UV disinfection. Both adaptation mechanisms have been considered, e.g. Kramer and Magee's widely adopted SAS architecture [5] has separate layers dedicated to configuration *change management* and *goal management*. However, most research to date has focused on the former layer, leaving the adaptation opportunities provided by goal management underexplored and poorly supported by existing modelling paradigms and adaptation methods.

This is a major limitation of current SAS, as indicated by recent research into human goal management [6]–[9], which shows that goal evolution is key to overcoming uncertainty and disruption. To address this limitation, we identify *desiderata* (i.e. high-level requirements) for a modelling paradigm to support goal adaptation (Sect. II), we propose an extended goal modelling (EDGE) notation that satisfies these desiderata (Sect. III), and we introduce a method for the formal synthesis of discrete-event controllers for the goal management layer of SAS (Sect. IV). We illustrate the application of our EDGE goal notation and controller synthesis method using a robotic SAS adapted from [10], we compare our approach to related work in Sect. V, and we summarise our future plans in Sect. VI.

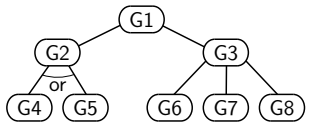

Fig. 1. Goal model comprising a root goal G1 that is achieved when subgoals G2 and G3 are both achieved; G2 is achieved when at least one of G4 and G5 is achieved, and G3 is achieved only when G6, G7 and G8 are all achieved.

II. Desiderata for SAS Goal Modelling

In goal-oriented requirements engineering (GORE) [11], [12], system goals are partitioned into subgoals using AND and OR decompositions. AND-decomposed goals (e.g., goal G1 in Fig. 1) are achieved if and only if all their subgoals are achieved, whereas OR-decomposed goals (e.g., goal G2 in Fig. 1) are achieved when one or more of their subgoals are achieved. Although SAS goal modelling research [13]–[22] has extended GORE with many useful features, these extensions (which we overview in Sect. V) cannot fully capture all aspects of a system's goals that—according to recent human goal management theories [6]–[9]—should be considered when dynamically selecting the goals to be pursued by a self-adaptive agent. As such, the full potential of SAS goal management can only be achieved by further extending existing goal modelling notations. To guide the development of these extensions, we used the aforementioned theories [6]–[9] to identify the following desiderata that they need to satisfy.

D1 Support for non-idempotent goal variants and their operationalisation – A vital strength of human adaptation is our ability to achieve high-level aims by dynamically selecting from multiple variants of the goals that support the realisation of these aims [9]. To reach a destination on time, we can accomplish the goal of reaching the train station by walking, cycling or taking the bus, and the goal of travelling to our destination by taking a fast or a slow train. Which variant of each goal we select depends on factors such as the weather, how tired we are, and which train service is running on time (operationalisation). Moreover, we *dynamically readjust our selection* to mitigate disruptions such as traffic jams and train cancellations. We argue that the SAS goal management

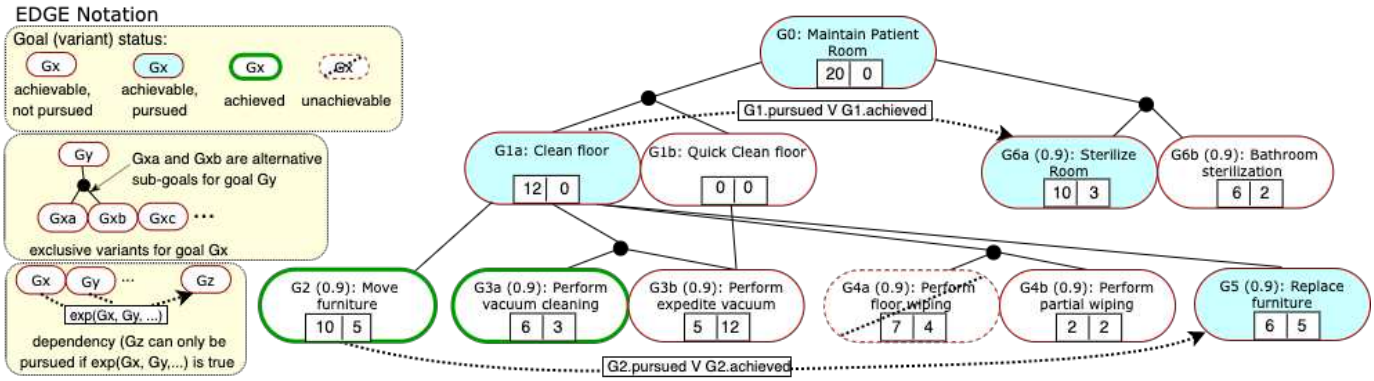

Fig. 2. EDGE notation (left) and goal model for a robotic SAS required to carry out maintenance of a patient room (right)

layer can only achieve its full potential by reasoning over goal models in which such *non-idempotent* goal variants are captured explicitly and dynamically selected whenever failure or environment constraints render some of the goal variants unachievable. In those cases, alternative non-idempotent goal variants are dynamically pursued to deliver an equally acceptable or degraded functionality, possibly with a different cost, level of risk, etc. To represent such goal variants, one could use the *alternative* node notation (|) from Dalpiaz et al. [23]. However, unlike [23], our desideratum does not require that all alternative (non-idempotent) goal variants are achievable when a goal controller is synthesised or used for the goal management SAS layer.

D2 Support for goal properties – To select the goal variants to pursue, humans consider the *properties* of the alternatives they choose from. Expected reward (or utility) is one such property widely confirmed by psychologists [6]–[8]. However, many other properties may influence our selection of a goal variant over another, including risk, cost and environmental impact. We posit that goal models should support a similarly sophisticated range of properties. This feature is essential to enable goal controllers to select combinations of goal variants that are both achievable and aligned with the SAS aims.

D3 Support for tracking goal status – As humans, (i) we know that not all goals are *achievable* at all times; (ii) we are aware which goals we are *pursuing*; (iii) we decide whether to pursue a goal based on our perceived *likelihood* of being able to achieve that goal; and (iv) we can ascertain whether a goal has been *achieved*. We argue that these four key properties need to be built-in goal properties, allowing goal controllers to track and control the SAS progress with delivering its goals. In our envisaged SAS goal management approach, the goal *likelihood* will be provided by domain experts and/or learnt from SAS logs, the *achievable* and *achieved* properties will be tracked and updated by the change management SAS layer; and the goal management layer will react to such updates by appropriately adapting the *pursued* property of goals.

D4 Support for goal interdependencies – The goal variants we pursue are rarely independent of each other. In the earlier example, deciding to walk to the train station is likely to take longer than cycling or taking the bus, and may mean that only the fast train service can take us to our destination on time. To allow the dynamic selection of compatible goal variants, a goal modelling notation needs to support the specification of these (and of more complex) types of goal interdependencies.

D5 Support for automating goal selection – A key human trait is our ability to reason over (mental) goal models in order to select which goal variants to pursue, and to revise this selection when it becomes unfeasible, suboptimal or otherwise undesirable. The human reasoning underpinning these processes is still the subject of intense research [6], [8], and thus unlikely to be fully elucidated and replicable by SAS any time soon. Nevertheless, we advocate the use of principled reasoning to *automate the synthesis of the logic* implemented by goal controllers. This asks for techniques for *mapping extended goal models to a formal representation* that formal controller-synthesis methods can operate with.

III. EDGE Notation for SAS Goal Management

To satisfy the desiderata from Section II, we propose a (preliminary) extended goal modelling (EDGE) notation that augments the established GORE notation [11], [12] with the new features from Fig. 2. This figure depicts the EDGE goal model for a SAS adapted from [10] and comprising a robot whose top-level goal (G0) is to maintain a patient room in a hospital by cleaning its floor in one of two ways (goal variants G1a, G1b, cf. desideratum D1) and sterilizing the whole room or just its bathroom (goal variants G6a, G6b). Cleaning the floor thoroughly (goal variant G1a) requires moving some of the room furniture (goal G2), vacuum cleaning (goal variants G3a, G3b), wiping the floor (goal variants G4a, G4b), and replacing the furniture in its initial position (goal G5).

The in-built goal properties (desideratum D3) are specified in EDGE through using different graphical notation for goals that were achieved (e.g., goal G2 in Fig. 2), for achievable goals that are not pursued (G3b), for achievable goals that are also pursued (G5) and for goals that are not achievable (G4a), with the likelihood of leaf goals placed in brackets after their goal IDs (0.9 for all such goals from Fig. 2). The *values* of the "regular" goal properties (desideratum D2) are listed

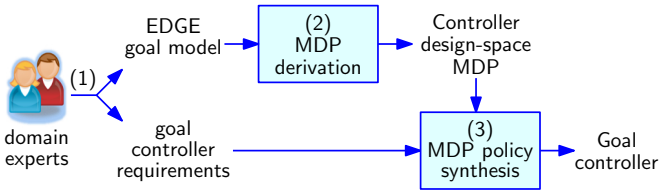

Fig. 3. EDGE goal controller synthesis: (1) the EDGE goal model and controller requirements (comprising constraints and a measure to optimise) are provided by domain experts; (2) an MDP that models the controller design space is derived through automated model-to-model transformation; and (3) the goal controller is synthesised through probabilistic model checking.

within the rectangular boxes included within each goal. Two real-valued properties are specified for each of the goals from our example: the *utility* and *cost* associated with achieving that goal. For instance, performing a thorough cleaning of the floor (goal G3a) is assigned a utility of 6 and a cost of 3; achieving goal G1a brings an additional utility of 12 on top the cumulated utilities of its sub-goals (because the maximal set of floor cleaning sub-goals was achieved) and has zero additional cost. We note that goal properties may also have other types, e.g., they can be of boolean type (like three of the built-in EDGE properties) or of categorical type.

Two goal interdependencies (desideratum D4) are shown in Fig. 1. First, sterilizing the whole room (goal G6a) should only be pursued if goal G1a is also pursued or was already achieved. Second, replacing the furniture (goal G5) only makes sense if the goal to move the furniture from its original position (goal G2) is pursued or was achieved. We explain how desideratum D5 is satisfied by EDGE in the next section.

IV. EDGE Goal Controller Synthesis

As shown in Fig. 3, EDGE goal models and goal controller requirements provided by domain experts (step 1 in Fig. 3) support the derivation of Markov decision processes or MDPs (step 2) *and* the synthesis of optimal MDP policies that define SAS goal controllers (step 3) guaranteed to: (i) satisfy a set of constraints over the goal properties; (ii) optimize a predefined measure over these properties. In the current EDGE version, the entire goal controller synthesis process is carried out at design time. As such, the way in which the synthesised controller responds to runtime events such as goals becoming unachievable, being achieved, etc. is decided beforehand.

The MDPs derived from EDGE goal models are specified in the modelling language of the probabilistic model checker PRISM [24], which models the behaviour of a system as the parallel composition of a set of *modules*. The state of a *module* is given by a set of finite-range local variables, and its state transitions are defined by guarded commands that change these variables, and have the form:

$$[action]\ guard \dashrightarrow e_1 : update_1 + \ldots + e_n : update_n;$$

where $guard$ is a boolean expression over all model variables. If $guard$ evaluates to $true$, the expression e_i, $1 \leq i \leq n$, gives the probability with which the $update_i$ change of the module

```
// Gx_pursued: 0 - not pursued, i>0 - pursued as variant i
module GoalController
  G2_pursued : [0..1] init 0; // G2 has one variant
  ...
  G6_pursued : [0..2] init 0; // G6 has two variants
  n : [0..5] init 0; // leaf goal counter

  // Decide whether to pursue goal G2
  [G2_skip] t & (n=0) & (G2_achieved | !G2_achievable) -> 1:(
        G2_pursued'=0)&(n'=n+1);
  [G2_pursue0] t & (n=0) & !G2_achieved & G2_achievable -> 1:(
        G2_pursued'=0)&(n'=n+1);
  [G2_pursue] t & (n=0) & !G2_achieved & G2_achievable -> 1:(
        G2_pursued'=1)&(n'=n+1);
  ...
  // Decide whether to pursue goal G6a|G6b
  [G6_skip] t & (n=4) & (G6a_achieved | G6b_achieved | (!(
        G6a_achievable & G1a) & !G6b_achievable)) -> 1:(G6_pursued'=0)
        &(n'=n+1);
  [G6_pursue0] t & (n=4) & !(G6a_achieved | G6b_achieved) & ((
        G6a_achievable & G1a) | G6b_achievable ) -> 1:(G6_pursued'=0)
        &(n'=n+1);
  [G6a_pursue] t & (n=4) & !(G6a_achieved | G6b_achieved) &
        G6a_achievable & G1a -> 1:(G6_pursued'=1)&(n'=n+1);
  [G6b_pursue] t & (n=4) & !(G6a_achieved | G6b_achieved) &
        G6b_achievable -> 1:(G6_pursued'=2)&(n'=n+1);
  // Controller done: inform Turn module and reset counter
  [controller_done] t & (n=5) -> 1:(n'=0);
endmodule

formula G1a = (G2_achieved | G2_pursued>0) & ...
                                & (G5_achieved | G5_pursued>0);
module ChangeMgmt
  G2_achievable: bool init true;   G2_achieved: bool init false;
  ...
  G6a_achievable : bool init true; G6a_achieved : bool init false;
  G6b_achievable : bool init true; G6b_achieved : bool init false;
  s : [0..6] init 0; // leaf goal counter
  fail : bool init false;

  // Outcome of pursuing goal G2
  [] !t & !fail & s=0 & G2_pursued=0 -> 1:(s'=s+1);
  [] !t & !fail & s=0 & G2_pursued>0 -> p2:(G2_achieved'=true)&(s'=s
       +1) + (1-p2):(G2_achievable'=false)&(fail'=true)&(s'=s+1);
  ...
  // Outcome of pursuing goal G6a|G6b
  [] !t & !fail & s=4 & G6_pursued=0 -> 1:(s'=s+1);
  [] !t & !fail & s=4 & G6_pursued=1 -> p6a:(G6a_achieved'=true)&(s'=
       s+1) + (1-p6a):(G6a_achievable'=false)&(fail'=true)&(s'=s+1);
  [] !t & !fail & s=4 & G6_pursued=2 -> p6b:(G6b_achieved'=true)&(s'=
       s+1) + (1-p6b):(G6b_achievable'=false)&(fail'=true)&(s'=s+1);
  // Plan failure: inform Turn module and reset counter
  [changeMgmt_done] !t & fail -> 1:(fail'=false)&(s'=0);
  // Plan success: move to end state
  [success] !t & !fail & s=5 -> (s'=s+1);
  [end] !t & s=6 -> (s'=6);
endmodule

module Turn
  t : bool init true; // true=controller, false=changeMgmt

  [controller_done] true -> 1:(t'=false);
  [changeMgmt_done] true -> 1:(t'=true);
endmodule

rewards "utility"
  [success] G0_achieved : 20;
  [success] G1a_achieved : 12;
  ...
endrewards

rewards "cost"
  [success] G0_achieved : 0;
  [success] G1a_achieved : 0;
  ...
endrewards
```

Listing 1. MDP derived from the EDGE goal model in Fig. 2

variables occurs. When $guard$ holds for multiple commands in a module, one of these commands (and its associated $action$) needs to be selected by a controller. This controller (also termed an *MDP policy*) is defined by the $action$ selected in each MDP state where multiple actions are possible. Finally, if multiple modules comprise commands with the same $action$, they must synchronise when performing this $action$, i.e., they must each perform one of these commands simultaneously.

As shown in Listing 1, which depicts the MDP obtained

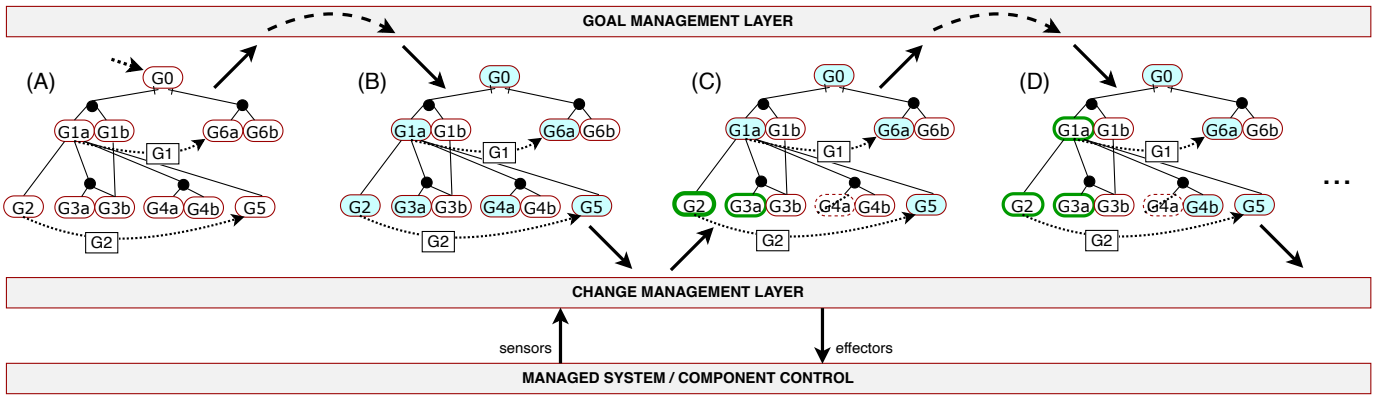

Fig. 4. Possible evolution of the EDGE goal model for the maintenance of a patient room: with all the goals initially achievable (A), the goal controller makes its initial goal selection (B), the change management layer issues an update to announce that goals G2 and G3a were achieved but G4a is unachievable (C), forcing the goal manager to select goal variant G4b as an alternative (D). Note that the evolving EDGE model is shown solely for explainability purposes – as the discrete-event goal controller responds automatically to events from the change management layer, the SAS does not need to store and update the EDGE goal model at runtime, although doing so can be useful for inclusion in a dashboard that shows the current status of the SAS to a human operator.

from the goal model in Fig. 2, EDGE-derived MDPs comprise three modules. First, the selection of the goals to be pursued by the SAS is modelled by a `GoalController` module. Second, the updates provided by the change management SAS layer are modelled by a `ChangeMgmt` module. Third, an auxiliary module `Turn` gives alternative "turns" to:

- the `GoalController`, for (i) selecting the initial set of goals to be pursued, and (ii) revising these goals after each goal property update by the `ChangeMgmt` module;
- the `ChangeMgmt` module, for updating the built-in goal properties *achievable* and *achieved* (and any regular goals properties that change and that the controller's decisions depend on) as required.

Every time when the module `GoalController` has a turn (i.e., when `t = true`), it examines each leaf goal from the EDGE model. If a goal is not achievable or was already achieved, the controller decides to not pursue it (actions `G2_skip` and `G6_skip` in Listing 1). Otherwise, the controller has the option to (still) not pursue the goal, or to pursue it in any available variant whose dependencies on other goals (if any, e.g., see how action `G6a_pursue` depends on formula `G1a` in Listing 1) are satisfied. The controller synthesis problem is to ensure that the right option is selected at each step where multiple options are available. After examining all leaf goals, the controller uses the action `controller_done` to notify the `Turn` module that its selection of goals (i.e., the *plan*) to be pursued by the SAS is complete.

The module `ChangeMgmt` models the way in which the SAS change management layer reports any goal variant found to be unachievable to the SAS goal management layer. To that end, `ChangeMgmt` encodes the outcome of pursuing each leaf goal variant from the plan in turn, starting with G2. If the goal is not pursued (i.e., if `G2_pursued=0`), the module moves to the next goal (`s'=s+1`); otherwise, the goal will either be achieved with probability $p2$ (specified as 0.9 in the EDGE goal model), or found to be unachievable with probability $1-p2$. In the latter scenario, the failure is recorded (`fail'=true`) and the action `changeMgmt_done` is triggered to notify the `Turn` module that goal controller must be invoked for the selection of an alternative plan. When no failures occur during the execution of the current plan, the `success` action and then the final `end` action are reached.

TABLE I
COMPARISON TO RELATED GOAL MANAGEMENT NOTATIONS

Goal modelling notation	Desiderata D1	D2	D3	D4	D5
AwReq [13], [14]	❍	◗	●		❍
GODA-MDP [15], [16]	❍	●	◗		◗
FLAGS [17]	❍	❍	◗		
RELAX [18], [19]	◗	◗	●		
RESPIRE [20], [21]	◗	◗	❍		
TROPOS4AS [22]	❍	◗	●		
EDGE	●	●	●	●	●

●=supported; ◗ = partly supported; ❍ = minimally considered

MDP rewards are used to assign the utility and cost values from Fig. 2 to the MDP states in which goal variants are achieved, and Fig. 4 depicts a possible evolution of the EDGE goal model for this SAS under a goal controller obtained by asking the model checker PRISM to synthesise an MDP policy that maximises the utility of the SAS without exceeding a cost of 25 (in this experiment, we set $p2 = \cdots = p6a = p6b = 0.9$ as in Fig. 2, and PRISM generated the MDP policy in 17.9s on a MacBook laptop with 2GHz Intel i5 processor and 32GB of memory). The complete MDP model and the other artifacts from this case study are provided in our GitHub repository [25] to allow the reproducibility of these results.

V. Related Work

While no existing goal modelling notation satisfies all the desiderata from Sect. II, several approaches provide at least partial support for some of them (Table I).

RELAX [18] supports the specification of environmental uncertainty in requirements, its AutoRELAX extension [19]

automates the generation of RELAX goal models and defines fuzzy logic function boundaries for goal satisfaction criteria, FLAGS [17] uses "adaptive goals" to define countermeasures that must be performed if one or more goals are not achieved, and Tropos4AS [22] supports the modelling of goal types with their associated satisfaction conditions. Each of these approaches fulfils to a limited extent desiderata D1, D2 and D3, but cannot synthesise SAS controllers for operationalizing more complex goal selection. The GODA-MDP framework [26] can model the probabilities of achieving the goals of a system, and has been used [15], [16] to develop SAS with control-theoretic/AI hybrid control loops. However, it only partially fulfills D1 and does not support D4 and D5, as it lacks constructs for the specification of non-idempotent goal variants and complex goal dependencies, and it produces SAS controllers that lack a separate goal management layer.

AwReqs [13], [14] can model the success or failure degree for requirements (in line with desideratum D1) and enables controller synthesis (D5), but cannot handle complex goal properties and dependencies. Finally, RESPIRE [20], [21] supports the specification of context/context-dependent goal properties, reasoning about changing environmental conditions, and the adaptation of goal models to these conditions. However, RESPIRE can neither model non-idempotent goal variants nor synthesise goal controllers, which are key desiderata for the goal modelling paradigm we propose in this paper.

VI. Conclusion

We used insights from human goal-management studies to identify desiderata for SAS goal management, introducing a goal modelling notation and a formal goal controller synthesis method that satisfy these desiderata. In future work, we plan to automate the MDP derivation step of our goal controller synthesis (Fig. 3), and to build on our recent research on multi-objective optimisation for probabilistic models [27]–[29] in order to support the synthesis of goal controllers that provide optimal trade-offs between multiple optimisation objectives. Additionally, we aim to extend our EDGE notation and controller synthesis with support for runtime changes to the goal likelihood and the EDGE model structure (enabling their use for SAS whose goal variants evolve over time), and for "continuous" goals, i.e., goals that a SAS must achieve over a period of time (e.g., maintaining the throughput of a server or the navigation speed of a robot within given limits). Finally, we plan to evaluate EDGE thoroughly using multiple case studies drawn from the repository of software engineering for SAS exemplars at [30] and from our recent work on goal-oriented approaches to SAS engineering [31]–[33].

Acknowledgements

This work has received funding from the UKRI project EP/V026747/1 'Trustworthy Autonomous Systems Node in Resilience', and the Assuring Autonomy International Programme. Genaína Rodrigues was also funded by the CNPq Call 04/2021 process 313215/2021-9.

References

[1] A. Khan and Y. Anwar, "Robots in healthcare: A survey," in *Science and Information Conference*. Springer, 2019, pp. 280–292.

[2] D. Lattanzi and G. Miller, "Review of robotic infrastructure inspection systems," *Journal of Infrastructure Systems*, vol. 23, no. 3, p. 04017004, 2017.

[3] Y. Brun, G. D. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering Self-Adaptive Systems through Feedback Loops," in *Software engineering for self-adaptive systems*. Springer, 2009, pp. 48–70.

[4] D. Weyns, *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley & Sons, 2020.

[5] J. Kramer and J. Magee, "Self-managed systems: An Architectural Challenge," in *Future of Software Engineering (FOSE'07)*. IEEE, 2007, pp. 259–268.

[6] F. Cushman and A. Morris, "Habitual Control of Goal Selection in Humans," *Proceedings of the National Academy of Sciences*, vol. 112, no. 45, pp. 13 817–13 822, 2015. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1506367112

[7] Q. J. M. Huys, N. Eshel, E. O'Nions, L. Sheridan, P. Dayan, and J. P. Roiser, "Bonsai Trees in Your Head: How the Pavlovian System Sculpts Goal-Directed Choices by Pruning Decision Trees," *PLOS Computational Biology*, vol. 8, no. 3, pp. 1–13, 03 2012. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1002410

[8] Q. J. M. Huys, N. Lally, P. Faulkner, N. Eshel, E. Seifritz, S. J. Gershman, P. Dayan, and J. P. Roiser, "Interplay of Approximate Planning Strategies," *Proceedings of the National Academy of Sciences*, vol. 112, no. 10, pp. 3098–3103, 2015. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1414219112

[9] A. Dezfouli and B. W. Balleine, "Actions, Action Sequences and Habits: Evidence That Goal-Directed and Habitual Action Control Are Hierarchically Organized," *PLOS Computational Biology*, vol. 9, no. 12, pp. 1–14, 12 2013. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1003364

[10] M. Askarpour, C. Tsigkanos, C. Menghi, R. Calinescu, P. Pelliccione, S. García, R. Caldas, T. J. von Oertzen, M. Wimmer, L. Berardinelli *et al.*, "RoboMAX: Robotic Mission Adaptation eXemplars," in *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2021, pp. 245–251.

[11] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, 2001, pp. 249–262.

[12] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models," in *International Conference on Conceptual Modeling*. Springer, 2002, pp. 167–181.

[13] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness Requirements for Adaptive Systems," in *2011 ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2011, Waikiki, Honolulu , HI, USA, May 23-24, 2011*, H. Giese and B. H. C. Cheng, Eds. ACM, 2011, pp. 60–69. [Online]. Available: https://doi.org/10.1145/1988008.1988018

[14] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, "Requirements-Driven Qualitative Adaptation," in *On the Move to Meaningful Internet Systems: OTM 2012*, R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. F. Cruz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 342–361.

[15] R. D. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. Vogel, and P. Pelliccione, "A Hybrid Approach Combining Control Theory and AI for Engineering Self-Adaptive Systems," in *SEAMS '20: IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Seoul, Republic of Korea, 29 June - 3 July, 2020*, S. Honiden, E. D. Nitto, and R. Calinescu, Eds. ACM, 2020, pp. 9–19. [Online]. Available: https://doi.org/10.1145/3387939.3391595

[16] A. Rodrigues, R. D. Caldas, G. N. Rodrigues, T. Vogel, and P. Pelliccione, "A Learning Approach to Enhance Assurances for Real-Time Self-Adaptive Systems," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 206–216. [Online]. Available: https://doi.org/10.1145/3194133.3194147

[17] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy Goals for Requirements-Driven Adaptation," in *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*. IEEE Computer Society, 2010, pp. 125–134. [Online]. Available: https://doi.org/10.1109/RE.2010.25

[18] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, "Relax: Incorporating Uncertainty into the Specification of Self-Adaptive Systems," in *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 2009, pp. 79–88.

[19] E. M. Fredericks, B. DeVries, and B. H. C. Cheng, "AutoRELAX: Automatically RELAXing a Goal Model to Address Uncertainty," *Empir. Softw. Eng.*, vol. 19, no. 5, pp. 1466–1501, 2014. [Online]. Available: https://doi.org/10.1007/s10664-014-9305-0

[20] D. Alrajeh, A. Cailliau, and A. van Lamsweerde, "Adapting Requirements Models to Varying Environments," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 50–61. [Online]. Available: https://doi.org/10.1145/3377811.3380927

[21] D. Alrajeh, P. Benjamin, and S. Uchitel, "Adaptation2: Adapting specification learners in assured adaptive systems," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1347–1352.

[22] M. Morandini, L. Penserini, A. Perini, and A. Marchetto, "Engineering Requirements for Adaptive Systems," *Requir. Eng.*, vol. 22, no. 1, pp. 77–103, 2017. [Online]. Available: https://doi.org/10.1007/s00766-015-0236-0

[23] F. Dalpiaz, A. Borgida, J. Horkoff, and J. Mylopoulos, "Runtime goal models: Keynote," in *IEEE Seventh International Conference on Research Challenges in Information Science (RCIS).*, May 2013, pp. 1–11.

[24] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-time Systems," in *CAV'11*, ser. LNCS, vol. 6806, 2011, pp. 585–591.

[25] "EDGE GitHub repository," 2023. [Online]. Available: https://anonymous.4open.science/r/Formalise23-05F8

[26] D. F. Mendonça, G. N. Rodrigues, R. Ali, V. Alves, and L. Baresi, "GODA: A Goal-Oriented Requirements Engineering Framework for Runtime Dependability Analysis," *Inf. Softw. Technol.*, vol. 80, pp. 245–264, 2016. [Online]. Available: https://doi.org/10.1016/j.infsof.2016.09.005

[27] R. Calinescu, M. Češka, S. Gerasimou, M. Kwiatkowska, and N. Paoletti, "Designing robust software systems through parametric markov chain synthesis," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 131–140.

[28] S. Gerasimou, R. Calinescu, and G. Tamburrelli, "Synthesis of probabilistic models for quality-of-service software engineering," *Automated Software Engineering*, vol. 25, pp. 785–831, 2018.

[29] S. Gerasimou, J. Cámara, R. Calinescu, N. Alasmari, F. Alhwikem, and X. Fang, "Evolutionary-Guided Synthesis of Verified Pareto-Optimal MDP Policies," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 842–853.

[30] "Software Engineering for Self-Adaptive Systems Exemplar Repository," 2022. [Online]. Available: https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/

[31] F. P. Guimarães, G. N. Rodrigues, R. Ali, and D. M. Batista, "Planning runtime software adaptation through pragmatic goal model," *Data Knowl. Eng.*, vol. 109, pp. 25–40, 2017. [Online]. Available: https://doi.org/10.1016/j.datak.2017.03.003

[32] G. F. Solano, R. D. Caldas, G. N. Rodrigues, T. Vogel, and P. Pelliccione, "Taming uncertainty in the assurance process of self-adaptive systems: a goal-oriented approach," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2019, pp. 89–99.

[33] G. S. Rodrigues, F. P. Guimarães, G. N. Rodrigues, A. Knauss, J. P. C. de Araújo, H. Andrade, and R. Ali, "GoalD: A goal-driven deployment framework for dynamic and heterogeneous computing environments," *Information and software technology*, vol. 111, pp. 159–176, 2019.