Computational Alignment of Goals and Scenarios for Complex Systems

Dalal Alrajeh and Alessandra Russo
Imperial College London
London, United Kingdom
{dalal.alrajeh, a.russo}@imperial.ac.uk

James Lockerbie and Neil Maiden
City University London
London, United Kingdom
{James.lockerbie.1, n.a.m.maiden}@city.ac.uk

Alistair Mavin
Rolls-Royce plc
Derby, United Kingdom
Alistair.Mavin@rolls-royce.com

Mark Novak
Aero Engine Controls
Derby, United Kingdom
mark.novak@aeroenginecontrols.com

***Abstract*—The purpose of requirements validation is to determine whether a large requirements set will lead to the achievement of system-related goals under different conditions – a task that needs automation if it is to be performed quickly and accurately. One reason for the current lack of software tools to undertake such validation is the absence of the computational mechanisms needed to associate scenario, system specification and goal analysis tools. Therefore, in this paper, we report first research experiments in developing these new capabilities, and demonstrate them with a non-trivial example associated with a Rolls-Royce aircraft engine software component.**

***Index Terms*—Requirements validation, scenarios, operational requirements, goal achievement.**

I. The Requirements Validation Challenge

The inability of software engineering projects to validate requirements so that new systems can achieve their high-level goals (e.g., [14]) has led to many high-profile failures, for example the NHS e-Records system (e.g., [4]). One reason for these failures is the lack of a cost-effective means with which to validate the emergent behaviour and qualities of a system for compliance with its goals. Whilst previous research has delivered elements of what is needed (e.g., automated scenario generation and systematic goal analysis tools), projects still lack the means with which to automatically generate the emergent behaviours and qualities of the specified system under different conditions expressed as scenarios, then analyze these behaviours and qualities for goal achievement. We argue that requirements projects increasingly need integrated tools with which to quickly generate large scenario sets, accurately compute future system behaviours from specified functions and systematically analyze goal achievement. Moreover, the tools need to be interactive so that analysts can explore the impact of different requirements configurations on goal achievement under different conditions.

As an example, consider variable stator vanes (VSVs) used in aircraft engines to change the incidence of airflow to the engine's rotor blades. The vanes are controlled by a system of sensors, electronic hardware and hydro-mechanical units to be specified. One validation task might be to investigate one scenario event – the impact of a missing speed signal between the intermediate-pressure shaft and the control system on the achievement of the goal that the torque motor continues running. New software capabilities are needed to integrate existing scenario, system specification and goal analysis techniques to enable such validation to take place. Figure 1 depicts the situation and the required new capabilities linking current tools.

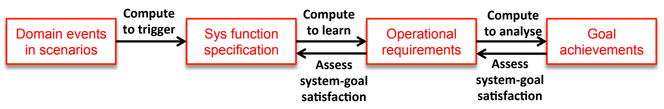
Domain events in scenarios
Compute to trigger
Sys function specification
Compute to learn
Assess system-goal satisfaction
Operational requirements
Compute to analyse
Assess system-goal satisfaction
Goal achievements

Figure 1: New capabilities linking existing scenario, system specification and goal analysis tools.

To deliver the new capabilities we have identified key research challenges that need to be overcome. These include:

1. How to predict system behaviour according to functions from different types of scenarios;
2. How to propagate the effect of requirements compliance on goal achievement;
3. How to assess subsequent system-goal satisfaction.

In this paper, we outline related work, and then report preliminary investigations on connecting goals and scenarios using logic programming analysis tools and learning new arguments for goal compliance. First findings are demonstrated with a complex requirements validation problem from our industrial partner Rolls-Royce.

II. Related Work

Surprisingly few computational mechanisms are available to generate and validate the emergent properties of a large set of specified requirements. From artificial intelligence, multi-objective evolutionary optimization algorithms have been applied to requirements selection in release planning [8], but these search-based techniques are limited to validating small numbers of predefined quality goals such as cost and time-to-deliver. Likewise, the KEYS2 Bayesian-based method has been combined with decision ordering diagrams to explore requirements-led changes [9] on single qualities such as robustness rather than a broader set of system and organizational goals. There is a need for new applications of artificial intelligence to analyse more complete sets of system behaviours and qualities, as well as enable human understanding of those behaviours and qualities.

Several machine learning-based approaches have been developed for goal analysis (e.g., [1]), requirements inference (e.g., [2,3,15]) and behaviour model synthesis (e.g., [6,7]). Though these methods produce sound results, they do not provide an automated mechanism for integrating domain assumptions, goals, requirements and abnormal behaviours and states each of which has been partially specified, nor for exploring the different, possible ways for integrating them.

Finally, most computational scenario techniques are used to specify required system behaviours rather than complex conditions in which the system must operate. For example, Live Statecharts extend UML message sequence charts with means for distinguishing between possible, necessary and forbidden behaviours [12] in tasks such as program synthesis [13]. To address these needs we are proposing new computational scenario techniques that can automatically generate different combinations of conditions under which systems must operate in forms that can exercise system specifications automatically.

III. Early Research Results

Our proposed work to overcome the research challenges builds on 3 pieces of research undertaken previously:

1. A computational tool for automatic scenario generation. The tool provides services with which to generate static scenario descriptions of possible inbound events that are deployed to validate requirements completeness against these events [18];
2. A tool-based framework that combines model checking, inductive learning and scenarios to support the elaboration of a complete set of operational requirements with respect to given set of functions [2];
3. A computational tool for automatic impact propagation in large goal models. The tool extends goal modelling in complex systems with goal satisfaction arguments to investigate the impact of atomic requirement changes on goal and requirement compliance [17].

To be able to predict system behaviour from different types of scenario design event, we are extending existing descriptive models of abnormal behaviour and state in scenario generation [18] with specifications of behavioural and quality consequences that can be implemented in computational form. To be able to reason about changes introduced to goal models, we are associating these with a declarative representation using logic programming. We are then using Answer Set Programming (ASP) [16] as a computational mechanism for aligning goals and scenarios, in order to explore the impact of changes to the goal model on goal achievement and to generate scenarios that illustrate the effect of such changes. Lastly, to be able to learn operational requirements from specifications of system functions and selected system qualities, and to propagate the effect of learned satisfaction arguments on goal achievement, we are extending the ASPAL declarative learning system [5] to accommodate probabilistic reasoning about complex system goal models.

IV. A Demonstrating Example

From the normal course specification of the VSV system provided by Rolls-Royce, we purposefully developed a goal model to demonstrate the following stages of our approach. Our aim is to implement these stages in an end-to-end requirements validation tool in which analysts can explore the effect of different specification configurations on goal achievement under different scenario conditions. As a first step we are prototyping each of the components reported in A-C separately to enable us to experiment with the potential benefits of the proposed new capabilities and tool integration.

A. Extending Goal Models with Inbound Scenario Events

Required system behaviours and qualities of a modified version of the VSV system are represented using the *i** model notation [19] shown in Figure 2. The model includes, in particular, relationships and goals not explicitly stated in the normal course description of the system, as the objective is to create the situation where changes are made to an existing (partial) model and impact analysis is then required.

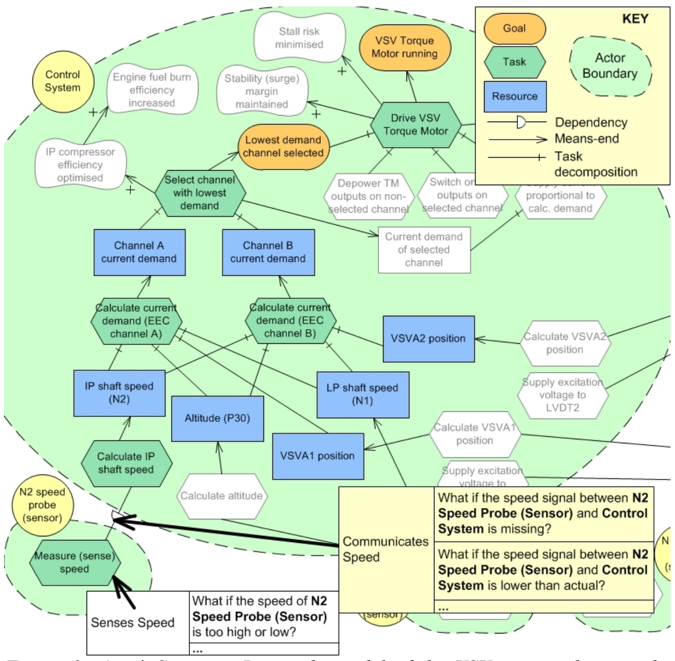
KEY
Goal
Task
Resource
Actor Boundary
Dependency
Means-end
Task decomposition
Control System
Engine fuel burn efficiency increased
Stall risk minimised
Stability (surge) margin maintained
VSV Torque Motor running
IP compressor efficiency optimised
Lowest demand channel selected
Drive VSV Torque Motor
Select channel with lowest demand
Depower TM outputs on non-selected channel
Switch on outputs on selected channel
proportional to calc. demand
Current demand of selected channel
Channel A current demand
Channel B current demand
Calculate current demand (EEC channel A)
Calculate current demand (EEC channel B)
VSVA2 position
Calculate VSVA2 position
Supply excitation voltage to LVDT2
IP shaft speed (N2)
Altitude (P30)
LP shaft speed (N1)
Calculate VSVA1 position
VSVA1 position
Calculate IP shaft speed
Calculate altitude
Supply excitation voltage to
N2 speed probe (sensor)
Measure (sense) speed
Communicates Speed
What if the speed signal between N2 Speed Probe (Sensor) and Control System is missing?
What if the speed signal between N2 Speed Probe (Sensor) and Control System is lower than actual?
...
Senses Speed
What if the speed of N2 Speed Probe (Sensor) is too high or low?
...

Figure 2: An i Strategic Rationale model of the VSV system showing the impact of selected abnormal scenario events on the completion of actor tasks and the validity of actor dependencies*

We build on the ART-SCENE tool to generate possible inbound scenario events to the VSV system. ART-SCENE deploys a tried-and-tested event generation rule-set, based on taxonomies of abnormal behaviour and state to generate possible alternative course events instantiated to given domains, such as *aircraft engines*. One such generated alternative course event is *what if the speed signal between two mechanical devices is missing?* One advance that we are experimenting with is to associate these events directly to goal models, for example the impact of a missing speed signal between the intermediate-pressure shaft sensor and the control system as expressed on the *i** model can be further explored.

B. Computing Candidate Impacts on Goal Achievement

We have chosen to use ASP as a means for generating impacts of possible *i** model changes arising from inbound scenario events on the achievement of goals that are specified in the model. ASP is particularly suited to support such task due to its efficient capacity to prompt and explore large search spaces in a timely manner. To deploy ASP, the *i** goal models need to be mapped into an ASP representation, whereby relationship between artifacts in the *i** model can be expressed as conjunctions of if-then rules.

Consider the goal model given in Figure 2. The following rules capture an excerpt of the model and in particular the goal that "*torque motor continues running*" (rule 1), tasks (rules 2-3), resources (rule 4) and links between them (rules 6-7).

```
goal(vsv_torque_motor_running).                                    (1)
...
task(drive_vsv_torque_motor).                                      (2)
task(calculate_demand_of_channel(a)).                              (3)
...
resource(ip_shaft_speed_n1).                                       (4)
channel(a).                                                        (5)
...
happens(achieve(vsv_torque_motor_running), N, S):-
   timepoint (N), scenario(S),                                     (6)
   holdsAt(achieved(drive_vsv_torque_motor), N, S).
happens(achieve(drive_vsv_torque_motor), N, S):-
   timepoint (N), scenario(S),                                     (7)
   channel(CH),
   holdsAt(achieved(select_channel_with_lowest_demand(CH), N, S),
   holdsAt(achieved(depower_tm_outputs_on_nonselected_channel,N,S),
   holdsAt(achieved(switch_on_tm_outputs_on_selected_channels,N,S),
   holdsAt(achieved(supply_current_proportional_to_calc_demand), N, S).
```

The predicates *goal*, *task*, *resource*, *channel*, *timepoint* and *scenario* are type predicates. For instance, rule (5) states that *a* is a constant of type channel. The semantics of the other predicates is given in Table I. In our interpretation, we consider achieve(T) and produce(R) to be actions that may be performed, and achieved(T) and available(R) to be fluents, i.e., time-varying properties that may be true or false. In this representation, variables are represented starting with an uppercase letter whilst constants with lower case ones.

TABLE I. DEFINITION OF PREDICATES USED IN ASP

Predicate	***Meaning***
achieve(T)	Achieve task T
achieved(T)	Task T is achieved
produce(R)	Produce resource R
available(R)	Resource R is available
happens(A, N , S)	Action A happens at time point N in scenario S
holdsAt(F, N, S)	Fluent F holds at time point N in scenario S

In essence, each rule codifies one link in the *i** model. For example, rule 1 states that the system achieves the goal "vsv_torque_motor_running" if the task "drive_vsv_torque_motor" is achieved, so capturing the means-end link between a goal and its associated task.

Our ASP representation provides a shared interface between the goal model and scenario-based specifications. Once it has been produced, we can use an ASP engine (e.g., *iclingo* [10]) to analyse goal achievement, compute scenarios allowed by the model and the impact of changes to the model on scenarios and goal achievement. We briefly describe our initial investigation of impact on goals using ASP.

A goal model can be verified for goal achievement by simply querying ASP whether all scenarios permitted by the ASP representation and a set of input entries achieve the goals. In our running example, the entries correspond to the *N2* and *N1 speed sensors measures*, *P30 engine pressure measure* and the *positions of VSVA1* and *VSVA2* as follows.

```
holdsAt(available(volt_prop_to_actuator_pos(vsva2)), 0, s1).      (8)
holdsAt(available(volt_prop_to_actuator_pos(vsva1)), 0, s1).      (9)
holdsAt(available(n1_speed_measure), 0, s1).                      (10)
holdsAt(available(n2_speed_measure), 0, s1).                      (11)
holdsAt(available(p30_engine_pressure_measure), 0, s1).           (12)
```

The tool then produces 32 varied scenarios in 0.010 second where the goal "vsv_torque_motor_running" is achieved.

Using the *what-if* statements produced by ART-SCENE, we can also explore the effects of exception cases on the achievement of the goals. For instance, consider the question "*What-if the speed signal between N2 speed probe sensor and the Control system is missing*" shown in Figure 2. To demonstrate the effect of this exception, we eliminate from our ASP representation the rule that represents the dependency link between the task "*Measure Speed*", controlled by the *N2 speed probe*, and the task "*Calculate IP shaft speed*" controlled by the *Control system*, i.e.,

```
happens(achieve(calculate_ip_shaft_speed), N, S):-
      timepoint(N),
      scenario(S),
      holdsAt(available(n2_speed_measure), N, S).
```

and run the tool. Consequently, the tool produces an output stating that the goal "vsv_torque_motor_running" can no longer be achieved. Furthermore, the tool can automatically generate scenarios explaining why the goal cannot be achieved. In our example, a sequence of tasks leading to undesirable states in which the tasks "*calculate_demand _of_channel(a)*" and "*calculate_demand_of_channel(b)*" can no longer be fulfilled as the required resource *IP shaft speed* is not available.

Our preliminary experiments suggest an increased benefit in using ASP as it allows us to predict system behaviour for given functions using different types of scenario design events and support related analysis of impacts on system-goals.

C. Learning Assumptions for Satisfaction Arguments

The semantics of the *i** model inherently represent some notion of satisfaction, for instance, a particular task has to be completed in order for its associated goal to be satisfied. In our example, for instance, the task of "*calculating current demand in channel a during steady state operation*" is satisfied by five resources being available, namely the *LP shaft speed*, *IP shaft speed*, *altitude* and the *positions for VSVS1* and *VSVA2*. However, the model itself does not include assumptions made by the engineer about the relationship between resources, tasks and goals, nor does it account for abnormal behaviours.

Satisfaction arguments [11] are used as a means of associating important domain assumptions to the specification of system behaviour leading to goal achievement. By attaching tasks and goals with satisfaction arguments, it is possible to determine whether a given high-level goal will hold or not. Our past experiences have revealed that satisfaction arguments are both time-consuming and difficult for human analysts to write. Therefore, the ability to generate all possible satisfaction arguments (i.e., to infer new ones or modify existing ones) is key to a successful alignment of goal models with scenarios. In the remainder of this section, we briefly describe how one aspect of the proposed learning capability link is the computation of assumptions for satisfaction arguments associated with tasks using results from Section IV.B.

To compute satisfaction arguments for the achievement of a task in a goal model, we need to identify a set of positive and negative scenarios to the goals' achievement in the original goal model. These scenarios may be either provided by the user or automatically generated using the ASP engine. Once identified, the user is expected to state which of the negative scenarios should not prevent the goals from being achieved, and therefore should not be considered as abnormal behaviour. The selected negative scenarios, called *accepted negatives*, are then augmented with the task for which the satisfaction argument is to be learned. Therefore by forcing the achievement of the task at the end of the negative scenario, we are suggesting that there is a domain assumption (or requirement) that is missing and is needed for the achievement of the task. The objective of the learning process is to find a new set of assumptions or to revise existing ones so that both positive and accepted negative scenarios are allowed to occur whilst satisfying the goals.

Referring back to our example, we use ASP to identify positive scenarios to the goal "vsv_torque_motor_running". To identify negative scenarios, we prompt ASP to generate scenarios in which the same goal is not achieved. For this, we use the same scenario produced in the previous subsection where the task "*calculate_demand_of_channel(a)*" and "*calculate_demand_of_channel(b)*" have both failed. We then augment the scenarios with desired tasks.

An ASP representation of the positive and accepted negative scenarios are given as examples to the learning tool ASPAL along with the original goal model. The outcome of this is an amendment to the implicit satisfaction argument for the concerned tasks stating that if the *IP shaft speed* resource cannot be produced, then the current demand for channels *A* and *B* are calculated using only the *LP shaft speed*, *altitude* and *VSVA1 position* (and *VSVA2 position for channel B).*

V. Discussion and Conclusion

Many technical challenges need still to be addressed to realize the new capabilities depicted in Figure1. Specifying an ordering over tasks in the goal model is important in order to map them into a scenario-based specification (e.g., MSC). Providing automated translation of the ASP output into MSC and vice versa is also essential in order to provide feedback to the analysts and requirements engineers. The preliminary investigation will need to be extended to consider the integration of operational requirements and complete satisfaction arguments, and the modelling and reasoning about a set of software qualities or soft goals. We believe that the declarative features of the existing analysis of learning tools that we aim to integrate will be key to the success of this ambitious research agenda.

Acknowledgment

This work is partially funded by the ERC project PBM–FIMBSE (No. 204853).

References

[1] D. Alrajeh, J. Kramer, A. van Lamsweerde, A. Russo, and S. Uchitel."Generating obstacle conditions for requirements completeness," In 34th IEEE/ACM ICSE, 705–715, 2012.

[2] D. Alrajeh, J. Kramer, A. Russo and S. Uchitel, "Elaborating requirements using model checking and inductive learning," IEEE TSE, in press, 2012, doi (10.1109/TSE.2012.41).

[3] R. V. Borges, A. Garcez, L. C. Lamb, and B. Nuseibeh. "Learning to adapt requirements specifications of evolving systems," In 33rd IEEE/ACM ICSE, 856–859, 2011.

[4] S. Chapman. "Hospital boss lambasts NHS e-records system," 2009. http://www.computerworlduk.com/news/it-business/13355/ hospital-boss-lambasts-nhse-records-system/.

[5] D. Corapi, A. Russo, and E.C. Lupu. "Inductive logic programming as abductive search," In 26th ICLP, 54–63, 2010.

[6] C. Damas, B. Lambeau, and A. van Lamsweerde. "Scenarios, goals, and state machines: a win-win partnership for model synthesis," In 14th ACM SIGSOFT FSE, 197–207, 2006.

[7] M. Emmi, D. Giannakopoulou, and C. Pasareanu. "Assume-guarantee verification for interface automata," In 15th Intl Symp. on Formal Methods, 116–131, 2008.

[8] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren and Y. Zhang. "Fairness analysis in requirements analysis," In 16th IEEE Intl conf. on RE, 115–124, 2008.

[9] G. Gay et. al. "Finding robust solutions in requirements models", J. ASE, **17**(1), 87–116, 2010.

[10] M. Gebser et. al. "Engineering an Incremental ASP Solver," In 24th ICLP, 190–205, 2008.

[11] J. Hammond, R. Rawlings and A. Hall. "Will it work?" In 5th IEEE Intl Symp. on RE, 102–109, 2001

[12] D. Harel and P. Thiagarajan. "Message Sequence Charts", in UML for Real: Design of Embedded Real-time Systems,, Kluwer Academic Publishers, 75–105, 2004.

[13] D. Harel and I. Segall. "Synthesis from Scenario-based Specifications", J. of Comp. Syst. Sci. 78(3), 970–980, 2012.

[14] C.W. Johnson and C.M. Holloway. "Questioning the role of requirements engineering in the causes of safety-critical software failures," In 1st Institution of Engineering and Technology Intl Conf. on System Safety, 352–361, 2006.

[15] A. van Lamsweerde and L. Willemet. "Inferring declarative requirements specifications from operational scenarios," IEEE TSE, 24(12), 1089–1114, 1998.

[16] V. Lifschitz. "Answer set programming and plan generation," AI, **138**(1-2), 39–54, 2002.

[17] J. Lockerbie and N. Maiden. "REDEPEND: Extending *i** modelling into requirements processes," In 14th IEEE RE, 354–355, 2006.

[18] A. Mavin and N. Maiden. "Determining socio-technical systems requirements: Experiences with generating and walking through scenarios," In 11th IEEE RE, 213 – 222, 2003.

[19] E. Yu, P. Giorgini, N. Maiden and J. Mylopoulos, "Social Modeling for Requirements Engineering," Cambridge, MA: MIT Press, 2011.