A Framework for Requirements Analysis Using Automated Reasoning

David Duffy*, Craig MacNish, John McDermid, Philip Morris*
Department of Computer Science
University of York,
York YO1 5DD, UK
{dad,craig,jam,philipm}@minster.york.ac.uk

Abstract. The problem of analysing the effects of changing requirements imposes strict demands on system representations, particularly in safety-critical domains. We argue that solving this problem will require structured representations that highlight the interaction between requirements, and record the rationale for decisions made during the development process.

As a means of providing and analysing this information, we propose the use of a goal-oriented model for structuring requirements and the use of formal reasoning techniques to aid in the analysis of changes and their consequences.

1 Introduction

The need to analyse the effects of changing requirements adds to the demands on system representations. In addition to verifying the integrity of the initial system, we would like to have the capability to trace the implications of changes through the system, and also to reason back from desired outcomes to determine what changes could bring them about. These capabilities rely upon a representation that focuses on both the interactions between system requirements and the interactions between the system and the context in which it operates. For this analysis to be productive, we also need some way of assessing the susceptibility of individual requirements to change, and therefore some way of recording the rationale for the design decisions that led to those requirements. All of these issues are particularly important in saftey-critical domains where maintaining the integrity of the system is paramount.

In this paper we propose a framework, called *goal-structured analysis* (GSA), which addresses these needs. The framework is based on goal decomposition supported by automated reasoning. A distinguishing feature of our approach from previous goal-based approaches (see for example [1, 3, 10]) is that we link the evolution of the requirements and design through the development of goals and system models, reflecting the fact that, for embedded systems, requirements and

* Supported by the DTI/SERC "PROTEUS" PROJECT IED4/1/9304 under the Safety-Critical Systems Initiative.

design must evolve in concert. We also place a strong emphasis on formal properties, such as non-circularity of justifications. The reasoning aspect is achieved through a logical representation which avoids many of the complexities of previous proposals (see for example [4, 9]) by operating in tandem with informal descriptions.

The main benefits of our approach include:

- Informal descriptions of requirements and design decisions, along with supporting evidence, are recorded in a structured way.
- Formal (logical) statements are coupled with the informal descriptions, facilitating formal and mechanised analyses. This allows us to make guarantees about the integrity of the specifications. For example, we can ensure that system models are consistent and that all goals are satisfiable under appropriate conditions.
- Conditions are provided for incremental development of requirements which, if followed during the development process, guarantee the above properties in the final structure.

In the following section we provide an overview of the framework and describe the intuitions behind the components. This is followed in Section 3 by a simple example of (informal) system development, which is used throughout the remainder of the paper. Section 4 discusses the addition of formal statements to the system description, while Section 5 defines some of the properties we would like to attain, and the conditions for incremental development which will bring these about. In Section 6 we give a very brief illustration of the way in which the framework allows us to predict system behaviour under different scenarios and examine changes to the system to improve its behaviour. Finally in Section 7 we summarise the work presented and discuss some future research directions.

2 Overview of the Framework

The fundamental components of our approach are goals, effects, facts and conditions, each containing both informal and formal information. In this section we briefly describe each of these components and their relationships, the types of information they contain, and the way in which this information is stored. In later sections we give a more rigorous account of the formal relationships between these components.

2.1 Goals and Effects

A *goal* contains a statement, or assertion, identifying a desired property of a (proposed) system. Goals are intended to be more flexible than (the traditional view of) requirements, which are often used to set contractual boundaries between customer and designer. The statements may express requirements that the system

should fulfil, derived requirements which emerge because of higher-level design decisions, or constraints (for example to do with cost or power consumption).

We associate with each goal *strategies* describing how that goal may be achieved. Alongside the strategies we record information that might be significant if changes are to be made at a later stage, such as the reason that the strategies are believed to be successful, criteria for choosing amongst strategies, and a justification for the selection of a particular strategy. A strategy will generally introduce further goals (along with effects, facts and conditions) leading to a hierarchy or, more generally, a network which we call a *goal structure*.

An *effect* is similar to a goal in that it makes a statement that requires further decomposition or explanation. The difference is that effects describe properties of the system which we might not necessarily *want* to achieve. From a decompositional point of view goals and effects are treated in a similar manner, and we will simply refer to goals where no confusion arises.

2.2 Facts and Conditions, Models and Scenarios

Whereas goals contain statements about the system that we wish to decompose further, *facts* are statements that we take to be true. For example, they may be statements about properties of the system that are known to hold, or design commitments (such as a choice of technology).

Conditions, like facts, contain statements that we do not wish to decompose any further. Unlike facts, conditions may not always hold. As the name suggests, we use conditions to analyse the behaviour of the system under different operating conditions. We call a (consistent) set of conditions a *scenario*.

We will see that all goals are eventually decomposed in such a way as to be supported or satisfied by the facts in some scenario. In this way, the set of facts can be regarded as a *model* of the system being developed, a scenario represents the input to that model, and the goals represent requirements on the output, or behaviour, of that system. This relationship is illustrated in Figure 1. The model will be developed or expanded in parallel with decomposition of the goals.

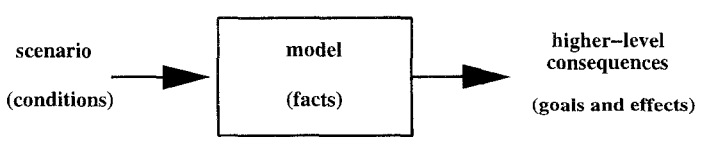

Fig. 1. Relationship between facts, conditions and goals.

2.3 Systems and Contexts

A new system is rarely developed in isolation, but rather is developed in the context of available components, subsystems, and the environment in which it

operates. This might include the "physical" world, a commercial environment such as the stockmarket, or a previously developed software system such as an operating system. In safety-critical systems constraints will also typically be imposed by regulatory bodies.

It is often convenient to separate the statements referring to different systems under consideration. For this reason we allow system labels. Furthermore any particular system under consideration may involve relationships with only a subset of the other systems being modelled. We call these other systems the *context* of the system under consideration. This modular view of systems and their associated contexts helps to address problems of scale.

2.4 Frame Representation

A natural representation for goals, effects, facts, conditions and their associated information is a *frame*, a data structure commonly used for representing hierarchical information in AI (see for example [7]). The following table shows some of the slots needed in a goal frame and the information envisaged for each slot.[2]

Slot	Information
Label	*identifier for the frame*
System	*system identifier*
Context	*identifier of directly related systems*
Assertion	*goal to be achieved or explained*
Alternatives	*list of alternative strategies*
Rationale	*reasons for belief in each alternative strategy*
Selection	*the chosen strategy*
Justification	*reason for selection of chosen strategy*

We concentrate our attention on these slots for the remainder of the paper. Examples are given in the following section.

Fact, effect, and condition frames are essentially cut down versions of goal frames. The selection slot in an effect frame will contain will contain an explication of the effect, and no alternatives slot will generally be needed. Since facts and conditions are not decomposed no strategy information is needed. However, the Rationale slot may be used to store the reason for believing in the fact or condition.

For brevity we have shown only one slot for each type of information, but in general many entries will involve two slots, one for an informal description and one for a formal description. The informal component allows the developers to describe the system in natural or semi-structured language. The formal component is used to guarantee properties of the system and to permit automated analysis. Note that both components need not necessarily be filled by the same engineer — for example the first may be the responsibility of the system designer while the second may contain a formalisation of this information by a specialist in formal languages.

[2] In practice a strategy will need to provide a solution to multiple goals (including making trade-offs between those goals) and may be shared between frames. For ease of exposition, however, we only associate strategies with single goals.

3 Buiding Goal Structures — An Example

In order to illustrate the construction and use of goal structures we will consider a simplified landing control system for an aircraft. The motivation for this example comes from various accounts concerning an landing incident (see for example [6]), but the details given here are contrived purely for the purposes of the example.

Briefly, the example involves the development of an automated system for applying wheel brakes and reverse thrusters with the aim of stopping an aircraft within 1000m of landing. The controller will make decisions about applying the brakes and reverse thrusters on the basis of measurements of wheel speed, load and altitude. These in turn will depend on environmental conditions when the aircraft touches down and the procedures followed by the pilot. We wish to analyse the behaviour of the aircraft under different scenarios involving these conditions.

There are four different 'systems' involved in the development: the landing control system itself, the dynamics of the aircraft, the operating procedures, and the environment in which the landing takes place. In this case, the latter three systems will provide the context in which the landing control system operates, and will mainly provide facts and conditions, since we are less concerned with interactions within those systems.

3.1 Developing the Landing Control System

We now run briefly through the development of the goal structure and system models. This allows us to illustrate how strategies may be derived from, or generate, facts and goals. For clarity we will consider only informal information at this stage, and show only those slots in the frames that are of principal interest. The corresponding formal information is detailed in the following section.

Deriving Strategies from Facts (or Goals) In the first step we give an example of the generation of a strategy from known information. We take our top-level goal to be to stop the aircraft within 1000m. (Note that in practice there will be other top-level goals such as those relating to safety.) The options available for the control system involve invoking wheel brakes and reverse thrusters, and an appropriate strategy must be developed with reference to the physical properties of the aircraft. We will assume that these physical properties have already been assessed in the process of designing, prototyping and/or testing the aircraft, and that this is reflected by the following facts:

Fact	
Label	F1
System	Aircraft dynamics
Assertion	The aircraft will stop within 1000m if wheel brakes are applied when speed $\leq$ 154km/h.
Rationale	Engineering tests

Fact	
Label	F2
System	Aircraft dynamics
Assertion	The aircraft will stop within 1000m if wheel brakes and reverse thrusters are applied when speed $>$ 154km/h and $\leq$ 170km/h.
Rationale	Engineering tests

These facts clearly suggest a strategy for satisfying our high-level goal. Since we wish to satisfy the goal for both ranges of speed, the facts are combined into a single strategy:

Goal	
Label	G1
System	Landing control
Context	Environment, Aircraft dynamics, Operating procedures
Assertion	Stop aircraft within 1000m
Selection	Apply wheel brakes when speed $\leq$ 154 knots, apply wheel brakes and reverse thrusters when speed $>$ 154 knots and $\leq$ 170 knots
Rationale	Based on F1, F2

The rationale for this strategy provides a pointer to the facts upon which it was based, thus recording the reason that the strategy is expected to be successful. Note that the assertions in the above facts may themselves appear as goals in the process of designing the dynamics of the aircraft. In future we will omit the Context slot as this will be inherited by other frames in the same system.

The strategy for G1 refers to four pieces of information. The application of the wheel brakes and reverse thrusters will be treated as new goals, while the statements concerning landing speeds (which result from the landing procedures and environmental conditions) will be treated as effects. Each of these can be decomposed further. In the next step we provide an example in which there are no predetermined strategies, and a design decision is made which in turn leads to a system description.

Generating Facts or Goals from Design Decisions In satisfying the goal of applying the reverse thrusters during landing, we will assume the designer has a choice of two strategies used in previous models, and makes a decision based on a safety constraint:

Goal	
Label	G2
System	Landing control
Assertion	Apply reverse thrust
Alternatives	Wheels loads $>$ 12 tonnes; altitude $<$ 10m and wheel speed $>$ 72km/h
Rationale	Both strategies have been used in previous models
Selection	Wheel loads $>$ 12 tonnes
Justification	Wheel loads considered more reliable in preventing premature application of reverse thrusters

For completeness of the goal structure, the causal relationship between the selected strategy and the goal must now be treated as a new goal or, in the case where we do not wish to decompose this any further, as a fact. For example:

Fact	
Label	F3
System	Landing control
Assertion	If wheel loads $>$ 12 tonnes then reverse thrust is applied
Rationale	Design decision from G2

Again the rationale contains a reference, this time from the fact to the goal, thus linking this "requirement" to the design decision upon which it is based.

The goal and fact frames for the application of the wheel brakes can be generated in a similar manner. For the sake of brevity we assume there is only one (disjunctive) alternative for the wheel brakes, leading to the following frames:

Goal	
Label	G3
System	Landing control
Assertion	Apply wheel brakes
Selection	Wheel loads > 12 tonnes or altitude < 10m and wheel speed > 72 km/h

Fact	
Label	F4
System	Landing control
Assertion	If wheel loads > 12 tonnes, or altitude < 10m and wheel speed > 72 km/h, then wheel brakes are applied

The above strategies introduce three new subgoals concerning the wheel load, wheel speed, and aircraft altitude, in addition to the two describing approach speed from G1. Apart from the altitude these subgoals can all be decomposed further by considering the dynamic properties of the aircraft.

Further Goals and Facts We will assume that the loading on both wheels depends on the way in which the aircraft is brought down. For simplicity we will say that the wheels are adequately loaded if there is a cross wind and the aircraft is banked, or if there is a tail wind and the aircraft is brought in level:

Fact	
Label	F5
System	Aircraft dynamics
Assertion	Wheel loads > 12 tonnes if the aircraft is banked when there is a cross wind or the aircraft is level when there is a tail wind

Goal	
Label	G4
System	Operating procedures
Assertion	Wheel loads > 12 tons
Selection	Bring the aircraft in banked in a cross wind and level in a tail wind

For simplicity we will assume the wheel speed is always greater than 72 km/h after touch down (a more sophisticated model might take account of conditions such as aquaplaning) and represent this with a fact F6.

The approach speed of the aircraft depends similarly on environmental conditions and the operating procedures. For example:

Fact	
Label	F7
System	Aircraft dynamics
Assertion	In a cross wind high throttle leads to a speed $\leq$ 154 km/h

Fact	
Label	F8
System	Aircraft dynamics
Assertion	In a tail wind high throttle leads to a speed > 154 km/h and $\leq$ 170 km/h

These facts provide explanations for the following effects:

Effect	
Label	E5
System	Operating procedures
Assertion	speed $\leq$ 154 km/h
Selection	Results from high throttle in a cross wind
Rationale	F7

Effect	
Label	E6
System	Operating procedures
Assertion	154 km/h < speed $\leq$ 170 km/h
Selection	Results from high throttle in a tail wind
Rationale	F8

Finally, we will assume that we are only interested in landing and therefore the altitude will necessarily be less than 10m, which is represented by a fact F9.

Conditions Describing a Scenario We are now left with subgoals referring to the operating procedures and weather conditions which will vary with different scenarios. For example, under normal landing conditions we might have:

Condition	
Label	C1
System	Environment
Assertion	Cross wind

Condition	
Label	C2
System	Operating procedures
Assertion	Bank aircraft and use high throttle

We leave the reader to verify informally that the aircraft lands successfully under these conditions, and return to this example in the following section.

3.2 Models and Support for Goals

During the construction of a goal structure we make use of existing descriptions of how components of the systems work, as well as generating new requirements for components. These are reflected in the facts, which form our model of the systems, and the conditions which describe the scenario in which they operate.

In each case where we specified a strategy for a goal (or explication for an effect) we "satisfied" ourselves that the strategy, along with other goals and facts expressed in the model, meant that the goal would be satisfied. We refer to this as *local support* for the goal, "local" because there is an assumption at this stage that any other goals referred to can be achieved through further decomposition. In some cases, where a fact directly expressed an implication from the subgoals in the strategy to the goal, it follows in a straightforward way that satisfying the subgoals would, in turn, satisfy the goal. In other cases a number of facts may have been required to ensure that the strategy led to the goal being satisfied.

As the development of a goal structure continues, any subgoals in the strategy that are not themselves facts are decomposed further. If this process is continued, with each decomposition being locally supported, until there are no further subgoals to be reduced, *every goal in the structure will be supported by the model alone.* In this case we say that all the goals are *globally supported.* It can be seen, therefore, that the strategies provide a means of developing, through a series of incremental steps, a model in which all the goals are satisfied.

Finally, we can investigate the response of the systems under various scenarios, as well as various changes to the systems, by following the implications of the changes through these "paths" of support.

In the remainder of the paper we formalise all of the above ideas by attaching logical descriptions of goals, facts and strategies to frames, and using formal proof methods to investigate and confirm local and global support for goals.

4 Formal Information in Goal Structures

In order to ensure properties such as consistency of our model and support for our goals, we need to have unambiguous representations of the assertions in goals, effects, facts and conditions, in a language that supports formal reasoning. We achieve this by adding logical information to our frames.

4.1 Examples of Logical Assertions

In this paper we will make use of only ground first-order sentences of a classical logic to describe assertions. (In general we may wish to use languages in which we can represent different kinds of causal relationships, numerical constraints, precedence of requirements and so on.)

As an example, the fact F1 might be expanded as follows:

Fact	
Label	F1
System	Aircraft dynamics
Assertion	The aircraft will stop within 1000m if wheel brakes are applied when speed $\leq$ 154km/h.
Assertion*	speed $\leq$ 154 $\rightarrow$ (applied(wheel.brakes) $\rightarrow$ stop.length $<$ 1000)
Rationale	Engineering tests

Assertion* contains the formal description of the contents of Assertion. We will sometimes use the notation $assert(\mathsf{F1})$ to refer to this slot directly. Similarly, if S is a set of frames we will write $assert(S)$ for the set of assertions therein.

In the case of a goal, as well as formalising the assertion made, we require a logical expression of the selected strategy. The reason for this is that, in the construction of a goal structure, the strategy temporarily takes the place of subsequent goals, effects, facts and conditions. This strategy also forms a semantic link between a goal and its "children". Thus G1 may be expanded to include:

Goal	
Label	G1
Assertion*	stop.length $<$ 1000
Selection*	[speed $\leq$ 154 $\wedge$ applied(wheel.brakes)] $\vee$ [154 $<$ speed $\leq$ 170 $\wedge$ applied(wheel.brakes) $\wedge$ applied(rev.thrust)]

Logical expressions can be similarly added to the other frames. Full lists for our example are shown in Table 1 and the two scenarios in Table 2.

As indicated earlier, whereas the informal descriptions will be provided by an engineer who has intimate knowledge of the design, the translation into logical form may be passed to someone with specific skills in logical representation and theorem proving. This is one of the motivations for combining formal and informal descriptions in this way.

5 Formal Properties of Goal Structures

Now that we have a formal description of the assertions and strategies in goal structures we are in a position to prove properties of the goal structure and the model. To simplify the exposition we will, where appropriate, refer to frames and their assertions interchangeably. For example, if G is a goal and M a set of facts, we will write $M \models G$ as shorthand for

$$\{assert(F) \mid F \in M\} \models assert(G).$$

Goals and Effects
G1 stop.length $<$ 1000
G2 applied(reverse.thrust)
G3 applied(wheel.brakes)
G4 wheel.loads $>$ 12
E5 speed $\leq$ 154
E6 154 $<$ speed $\leq$ 170

Model M1
F1 speed $\leq$ 154 $\rightarrow$ (applied(wheel.brakes) $\rightarrow$ stop.length $<$ 1000)
F2 154 $<$ speed $\leq$ 170 $\rightarrow$
(applied(wheel.brakes) $\wedge$ applied(reverse.thrust) $\rightarrow$ stop.length $<$ 1000)
F3 wheel.loads $>$ 12 $\rightarrow$ applied(reverse.thrust)
F4 wheel.loads $>$ 12 $\vee$ altitude $<$ 10 $\wedge$ wheel.speed $>$ 72 $\rightarrow$ applied(wheel.brakes)
F5 (cross.wind $\rightarrow$ (banked $\rightarrow$ wheel.loads $>$ 12)) $\wedge$ (tail.wind $\rightarrow$ (level $\rightarrow$ wheel.loads $>$ 12))
F6 wheel.speed $>$ 72
F7 cross.wind $\rightarrow$ (throttle(high) $\rightarrow$ speed $\leq$ 154)
F8 tail.wind $\rightarrow$ (throttle(high) $\rightarrow$ 154 $<$ speed $\leq$ 170)
F9 altitude $<$ 10

Table 1. Logical assertions corresponding to goals and facts.

Scenario W1	Scenario W2
C1 cross.wind	C3 tail.wind
C2 banked $\wedge$ throttle(high)	C2 banked $\wedge$ throttle(high)

Table 2. Logical descriptions of two scenarios.

5.1 Consistency and Global Support

First, we would like to ensure that our model is consistent, any scenario we apply the model in is consistent, and also that the union of any such scenario and the model is consistent (otherwise *any* goal assertion will be a consequence). These can all be tested using a standard theorem prover.

Secondly, we would like all goals (and effects) to be satisfiable in some scenario. This is the condition of global support introduced in Section 3.2, and ensures that the goals have been adequately (at least from a logical point of view) decomposed. We achieve this by ensuring that all goals are logical consequences of the union of the model and some scenario consistent with the model, as illustrated in Figure 2.

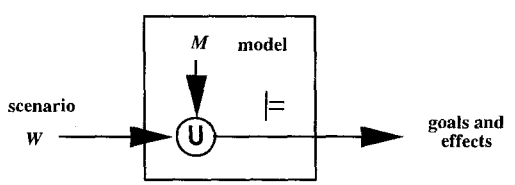

Fig. 2. Formal relationship between facts, conditions, goals and effects.

Definition 1 *Let G be a goal or effect in some context with model M. Then G is* (globally) supported *in a scenario W iff*

$$W \cup M \models G.$$

We say that a goal or effect is *supportable* if and only if it is supported in some scenario W such that $W \cup M$ is consistent.

In our example we can see that goals G1,...,G4 and effect E5 are consequences of M1 and W1, while E6 is a consequence of M1 and W2. Thus all goals and effects are supportable.

Like the consistency properties, global support can be established using a standard theorem prover once the goal structure has been completely decomposed. A more difficult task is to specify checks that can be made during the decomposition process to ensure that the final goal structure will be globally supported. That is, we wish to specify the formal conditions for local support.

5.2 Local Support

Recall that we would like a goal (or effect) to be locally supported if it is satisfied by other goals (intuitively those at the next level of decomposition) as well as facts and conditions. This includes subgoals mentioned in the strategy for which new frames have not yet been added. Thus if a context C includes model M, goals and effects H, and a scenario W such that $W \cup M \cup H$ is consistent, then for any particular goal (or effect) G with selection S, we would like to have:

$$assert(M \cup (H - \{G\}) \cup W) \cup \{S\} \models assert(G).$$

Note that we are assuming here that C and its components may not be completely decomposed. Providing we are dealing with a monotonic logic (as in this case) the above relationship will still hold as any further assertions are added to the left of the consequence relation.

This relationship would not, however, be sufficient to guarantee global support. The reason is that it allows circular paths of support. Consider, for example, a goal G_1 with assertion A and strategy B. This would be locally supported by another goal G_2 with assertion $B \rightarrow A$. However G_2 is also locally supported by G_1. In this case neither goal need be globally supported.

To overcome this problem we impose a partial order $\preceq$ (and corresponding quasi-order $\prec$) on assertions, and hence on the goals (effects) containing them. We then insist that the local support for a goal from other goals comes only from those that are strictly lower in the partial order. We also require that the goal's strategy is eventually entailed by goals that are lower in the partial order, along with facts and conditions. Since the partial order is transitive and antisymmetric this prevents the occurrence of loops.

If G is a goal in context C we use $\prec_G C$ to denote the set of goals in C that are strictly lower in the partial order than G; that is

$$\prec_G C = \{A \mid A \in C \text{ and } A \prec G\}.$$

We can then define local support recursively as follows:

Definition 2 *Let G be a goal or effect with strategy S and context C. Let M be the model in C and W a scenario in C such that $M \cup \prec_G C \cup \{S\} \cup W$ is consistent. Then G is locally supported in W iff there exists*

$$T \subseteq \prec_G C$$

such that

$$assert(M \cup T \cup W) \cup \{S\} \models assert(G) \quad (1)$$

and

$$assert(M \cup T \cup W) \models S \quad (2)$$

and the goals and effects in T are locally supported in W.[3]

The partial order will be constructed along with the goal structure. The general idea is that the assertions of any goals, facts or conditions that the developer makes use of to satisfy a goal must precede the goal in the partial order. Note that the strategy is provided for convenience and may be empty.

Theorem 3. *In a finite goal structure, any goal that is locally supported in some scenario is globally supported in that scenario. Hence any locally supportable goal is globally supportable.*

Proof. Let G, S, C, M, W and T be defined as in Definition 2. If T is empty, then G is trivially globally supported in W. Suppose then that T is non-empty. Since the goal structure is assumed finite and $\prec$ is an irreflexive partial order, $\prec$ is well-founded on the goal structure. Hence it may be assumed as an inductive hypothesis that the goals in T are globally supported in W. Thus, for any goal $J \in T$, we have $M \cup W \models J$; but then by equation (2) $assert(M \cup W) \models S$ and by equation (1) $assert(M \cup W) \cup \{S\} \models assert(G)$. It follows that $M \cup W \models G$ and thus G is globally supported in W. □

As an example, in Scenario W1, goal G1 is locally supported by G3 and E5 along with the model. G3 is supported directly by the model in W1, as is E5. Therefore by Theorem 3 G1 is globally supported in W1. E6, on the other hand, is locally (and globally) supported in W2.

6 Simple Analysis of Changes

Just as the logic-based framework allows us to reason about consistency and support, it allows us to assess the implications of various changes to the system. While we do not have space in this paper to address this issue in depth, the simple

[3] Note that from equations (1) and (2) it follows that $assert(M \cup T \cup W) \models assert(G)$. This is what we would expect — the strategy simply provides a convenient intermediate stage.

example we have set up allows us to give a flavour of how the system might be used.

In the incident described, it is assumed that the pilot is informed of a cross wind, and adopts the standard landing procedure of increasing the thrust and banking the aircraft — Scenario W1. Applying this to our model provides G1 (that is, the aircraft stops within 1000m) as a logical consequence as expected.

Just prior to the landing, however, the wind swings around to a tail wind — Scenario W2. As can be seen from the model, this causes a coincidence of two effects: first, the landing speed is increased so that both the reverse thrusters and the wheel brakes are required; and second, the banking of the aircraft means that the wheels are not both adequately weighted, so the reverse thrusters do not fire. Taken together these effects mean that the aircraft no longer stops within the required distance. Using a theorem prover we see that G1 is no longer a logical consequence of the model in this scenario.

While in this case it is not difficult to trace the implications of scenario W2 by hand (at least once the exercise of formalising the relationships has been performed) in a complex system this will not always be the case, increasing the benefits of automated impact analysis.

Finally, we would like to see what changes can be made to our system in order to prevent this incident occurring. Here the logical model can be used for "what if" analysis. In this example there is a simple change which prevents the accident — the alternative strategy is chosen for goal G2. Modifying fact F3 accordingly and resubmitting the second scenario to the proof system verifies that the problem no longer occurs.

7 Conclusions and Future Work

Goal-structured analysis provides a methodology for structuring system requirements, underpinning these with logical representations, and conducting automated analyses of the resulting systems.

The framework we have presented here is restrictive and further work is needed from both theoretical and pragmatic points of view. This work is continuing with reference to case-studies, carried out with industrial partners on the project, to ascertain what information it is most important to represent and what functionality is required for analysing this information. At present we are investigating adding the following facilities:

- A *suggestion* facility, based on abductive reasoning (e.g. [2]), that proposes ways of developing the goal structure to support goals which do not hold.
- A *sensitivity* facility that uses labelled deduction [5] to partition out goals which cannot be affected by changes to specific parts of the system, avoiding reprocessing.
- A *selection* facility, based on prioritized logics (see for example [11]), that allows requirements to be ranked in terms of the difficulty of change, and enables the reasoning system to select candidates accordingly.

- An *information extraction* system that assists the user in extracting logical information from the informal descriptions expressed in controlled natural langauge [8].
- An *explanation* facility that conveys to the user the line of reasoning used in deductions.

While more research needs to be carried out to ascertain the feasibility of these facilities, we believe that the framework described in this paper provides a useful platform for investigating these areas.

Acknowledgements

We would like to thank the academic and industrial partners on the Proteus project for valuable discussions.

References

1. BARBER, G. Supporting organizational problem solving with a workstation. *ACM Trans. on Office Information Systems 1*, 1 (1983), 45–67.
2. BONDARENKO, A. G. Abductive systems for non-monotonic reasoning. In *Logic Programming: First and Second Russian Conferences on Logic Programming, LNAI 592* (Berlin, 1992), A. Voronkov, Ed., Springer-Verlag, pp. 55–66.
3. DARDENNE, A., VAN LAMSWEERDE, A., AND FICKAS, S. Goal directed requirements acquisition. *Science of Computer Programming 20* (1993), 3–50.
4. DUBOIS, E. A logic of action for supporting goal-oriented elaborations of requirements. *5th International Workshop on Software Specification and Design, ACM Sigsoft Engineering Notes 14* (1989), 160–168.
5. GABBAY, D. Abduction in labelled deductive systems — a conceptual abstract. In *Symbolic and Quantitative Approaches to Uncertainty: Proc. European Conference ECSQAU. LNCS 548*, R. Kruse and P. Siegel, Eds. Springer-Verlag, 1991, pp. 3–11.
6. LADKIN, P. Analysis of a technical description of the Airbus A320 braking system. CRIN-CNRS & INRIA Lorraine, BP 239, Vandoeuvre-Lès-Nancy, France.
7. LUGER, G. F., AND STUBBLEFIELD, W. A. *Artificial Intellegence: Structures and Strategies for Complex Problem Solving.* Benjamin Cummings, 1993.
8. MACIAS, B., AND PULMAN, S. Natural language processing for requirements specifications. In *Safety-critical Systems*, F. Redmill and T. Anderson, Eds. Chapman and Hall, 1993, pp. 67–89.
9. MAIBAUM, T. A logic for the formal requirements specification of real-time embedded systems. Tech. rep., Dept. of Computing, Imperial College, London, 1987. Forest Deliverable R3.
10. MYLOPOULOS, J., CHUNG, L., AND NIXON, B. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Trans. on Software Engineering 18*, 6 (1992), 483–497.
11. RYAN, M. *Ordered Presentations of Theories — Default Reasoning and Belief Revision.* PhD thesis, Dept. of Computing, Imperial College, London, 1992.