Using Satisfaction Arguments with $i^*$ Modelling: An Exploratory Integration and Application to an Air Traffic Management System

Neil Maiden$^1$  James Lockerbie$^1$  Debbie Randall$^2$  Sean Jones$^2$  David Bush$^2$

City University London$^1$
Centre for HCI Design
London EC1V 0HB, UK
n.a.m.maiden@city.ac.uk

NATS Corporate & Technical Centre$^2$
4000 Parkway, Whiteley
Fareham, Hants PO17 7FL, UK
Debbie.RANDALL@nats.co.uk

Abstract
This technical research paper reports the integration of satisfaction arguments into the $i^*$ goal modelling approach to support analyses of the impact of new software systems on system-wide goals. Integration is based on a model that relates satisfaction argument and $i^*$ model concepts. New procedures to use the models and satisfaction arguments are supported within a software tool for $i^*$ modelling. Results are demonstrated using models and arguments developed for a new software tool with which to detect controlled airspace infringement in the United Kingdom. Preliminary results from an evaluation of the tool and procedures are reported.

1. Introduction
Analysts are increasingly using $i^*$, the strategic goal modeling approach [16], to model and analyze requirements. $i^*$ has been applied successfully to model requirements for air traffic management tools [8, 10] and decision support aids in agriculture [12] as well as to support individuals and groups in the work of charitable organizations [13]. Reported benefits to our projects have included automatic requirements generation from $i^*$ models [10] and detection of omissions in UML requirements specifications [8]. However, our experiences with $i^*$ in these projects also revealed 2 important weaknesses: (i) inadequate semantics to express means-end links, and (ii) poor integration with in-house requirements processes. In this technical research paper we report research results that extended $i^*$ to overcome these 2 weaknesses.

The first research extension was to add satisfaction arguments from REVEAL [4] to provide additional semantics for modelling means-end links. Each satisfaction argument adds more information to a means-end link and associates it with important properties of the problem domain. The second research extension was to develop new procedures with which to exploit $i^*$ models more effectively in requirements processes. These procedures, and the tools developed to support them, were designed to enable analysts to explore the impact of new software requirements on system-wide goals such as safety. Proof of concept of both extensions was demonstrated with a requirements project undertaken at NATS, the UK’s national air traffic service. Requirements analysts used the extended version of $i^*$ to model safety-related goals associated with a new software tool called the Controlled Airspace Infringement Tool (CAIT). The new procedures were then applied to explore the impact of the new tool requirements on the safety-related goals of the wider air traffic management system.

The remainder of the paper is in 8 sections. Section 2 describes the 2 weaknesses with $i^*$ and outlines our solutions to them. Section 3 describes the airspace infringement detection problem to which NATS applied these solutions. Sections 4 and 5 detail the 2 solutions reported originally in [20], then section 6 reports how the solutions were implemented in our $i^*$ modelling tool and applied to CAIT. Section 7 briefly outlines initial evaluation results. Section 8 reviews the claims made for the reported research, and section 9 summarizes future work.

2. Using $i^*$ in Requirements Projects
$i^*$ is an approach with which to model information systems composed of heterogeneous actors with different, often competing goals that depend on each other to undertake their tasks and achieve these goals [16]. It is an established approach for goal modeling, and has given rise to different versions of $i^*$ syntax and semantics that support different styles and uses of $i^*$ modeling. Most versions support the development of 2 types of $i^*$ model.

The first type of $i^*$ model is the Strategic Dependency (SD) model. The SD model provides a network of dependency relationships among actors. The opportunities available to these actors can be explored by matching the depender who is the actor who “wants” and the dependee who has the “ability”. Since the dependee’s abilities can match the depender’s requests, a high-level strategic model can be developed.

The second type of $i^*$ model is the Strategic Rationale (SR) model. The SR model provides an intentional description of goal and task elements and the relationships linking them. An element is included in the SR model only if it is considered important enough to affect the achievement of some goal. The SR model includes the SD model, and hence actors in the SR model either accom-
plish something by themselves or depend on other actors. The SR model has 4 main types of element: goals, tasks, resources and soft goals. These 4 types can be linked using any of the 4 available relationships links available in the SR model: the dependency link, the task decomposition link, the means-end link and its specialization, the contributes-to soft goal link.

In our requirements projects we support i* modeling with a software tool called REDEPEND, which extends Microsoft Visio with features specific to i* to enable requirements analysts to model and analyze SD and SR models. It provides a graphical palette from which analysts can drag-and-drop then directly manipulate i* model elements. It also provides simple model analysis features to verify SD and SR models that, due to their size, are difficult to verify manually. Indeed, both direct manipulation and automated model verification are seen as essential for scaleable i* modeling.

In the last 6 years we have applied i* and REDEPEND to model requirements for 4 major air traffic management systems, including a departure management system for major European airports [8] and a system that supports the scheduling of UK airspace [9]. REDEPEND supports version 5.0 of i* tailored to RESCUE based on recommendations to improve the notation and tool features from analysts in these previous projects. It permits styles of modeling not supported in other i* versions, including means-end links across actor boundaries to support modeling of complex trade-offs in the development of socio-technical systems.

Whilst i* provided important new capabilities in these projects that we have reported elsewhere [10], the projects also revealed 2 weaknesses that, we argue, need to be overcome to ensure widespread industrial uptake of i*.

2.1 Encountered Weaknesses with i*

The first weakness is the i* means-end link. i* uses the means-end link to express how goals and soft goals are attained and achieved. Means-end links associate i* elements that represent features of a new solution – most often tasks and resources – with states that these actors want to attain or achieve – goals and soft goals. However, analysts in our previous projects, when expressing means-end links, reasoned about much more than just the means and the end. They also established a large number of assumptions that had to be true in order for the means to be a means to the end, and refined the degree to which means attained or achieved ends. Unfortunately, i* offered little to support and capture the results of such reasoning.

For example, in an earlier air traffic management project called VANTAGE [11], we produced an i* SR model showing how enhanced airport operations could minimise the environmental impact of a regional airport. The model includes a simple means-end link between the task continuous descent approach (CDA) undertaken by the pilot and the airport management soft goal aircraft noise minimised. It states that successful completion of the pilot task contributes positively to the airport management achieving the soft goal. However, justification of this means-end link in VANTAGE was more complex. Completing the task only contributed to achievement of the soft goal if certain assumptions were true, for example that the current approach procedures are indeed noisier than CDA, and that airspace can be reorganised in a way that does not require low-altitude manoeuvres.

Whilst i* already includes the concept of a belief, a form of claim applied to means-end links, they are intended to express conflict arising from actors’ different beliefs [15] rather than more general assumptions associated with the problem domain. Furthermore, in our projects, each means-end link was often associated with multiple assumptions. If all assumptions were modelled as beliefs using the i* notation (a cloud), the resulting model would be cumbersome to develop and unclear to read.

The second weakness is a lack of guidance on how to embed research-based techniques such as i* in requirements processes. NATS projects have characteristics common with requirements projects in many organisations. Analysts often write requirements on the new software system that is the focus of the project, rather than on the wider socio-technical system that the software is a part of. Analysts specify these requirements in text form using traditional system shall statements. The requirements tend to be functional rather than non-functional. And the requirements are often difficult to link using existing traceability techniques to system-wide concerns such as safety and security. NATS cannot overlook its established requirements processes to accommodate new i* modelling. Therefore we explored new ways of developing and using i* models in processes that express software requirements in text form.

We extended i* to address the 2 reported weaknesses. The first solution was to integrate satisfaction arguments into i*, and in particular to use satisfaction arguments to elaborate i* means-end links. The second solution was to develop a new procedure for using i* models during traditional requirements processes, and new software tools to support the procedure. Each is described in turn.

2.2 Satisfaction Arguments in i*

Satisfaction arguments were first introduced to recognize the role of domain knowledge in requirements specification [19] and were applied effectively in a simplified form in the REVEAL requirements method [4]. A satisfaction argument relates domain knowledge to the introduction of a system to satisfy requirements:

\[ D, S \models R \]

A satisfaction argument should read: using the relevant properties of the application domain (D), when combined with the specification of the behaviour of the system (S)
to be implemented, it is possible to show (\(|\) that the requirement \((R)\) will hold [4].

In REDEPEND we extended \(i^*\) means-end links with satisfaction arguments. An analyst can develop a satisfaction argument for any goal or soft goal that is the end of a means-end link in an \(i^*\) SR model. In simple terms the end-element describes the requirement \(R\) and each means-element linked to the end-element describes a specification \(S\). The satisfaction argument provides an argument for all of the means-end links associated with the end-element. Therefore requirements and specifications are relative to the satisfaction argument. A model element can be a means-element (specification) in one argument and an end-element (requirement) in another. In section 4 we describe how we support multiple satisfaction arguments using boundaries of socio-technical systems.

2.3 Using \(i^*\) Models to Explore How Software Requirements Impact Safety-Related Goals

Linking \(i^*\) models to text requirement specifications is not new. Indeed, in 2 previous ATM projects we generated candidate requirement statements from \(i^*\) models using a pattern-based approach [10]. However one limitation was that the \(i^*\) models had to be generated first. This was not possible in requirements processes in NATS. Therefore we developed a new procedure to explore the impact of documented software requirements on system-wide goals using \(i^*\) models. Our goal was that an analyst is able to assess the impact of the new software system on the attainment and achievement of actor goals and soft goals that, in CAIT, are safety-related.

To develop the procedure we developed a simple matrix with which to map software requirements to \(i^*\) SR model elements and heuristics to propagate and analyze the impact of missing or unsatisfied requirements on system-wide goals and soft goals in the SR model. We extended REDEPEND with new features that enable an analyst to complete the matrix and propagate the impact of missing or unsatisfied requirements. The procedure shares characteristics with the qualitative \(i^*\) evaluation procedure reported in [5] that applies propagation rules to \(i^*\) models to apply labels representing the level of evidence towards the qualitative satisfaction and denial of model elements. However the procedure reported in this paper, we conjecture, also links \(i^*\) model elements to software requirements and provides stronger guidance for analyzing means-end links using satisfaction arguments.

2.4 Research Contributions

We claim 4 novel contributions for the work that we report. The first is the conceptual integration of \(i^*\) SR models and satisfaction arguments. The second is to extend reported procedures that build on this integration to exploit \(i^*\) models to analyze the impact of specified software systems on system-wide goals. The third is a set of novel software features that extend SR models with satisfaction arguments, and support and exploit the new procedure. All are demonstrated using models and requirements developed for a complex air traffic management problem reported in the next section. Hence our fourth claim is that our tools and techniques scale to model large and complex socio-technical systems with \(i^*\).

3. CAIT: \(i^*\) Modelling of a Complex Air Traffic Management System

In the United Kingdom airspace is broadly divided into 2 types; controlled and un-controlled airspace. Aircrew must obtain air traffic control clearance prior to entering controlled airspace. However, pilots are not always aware where they are, and the number of reports of aircraft entering controlled airspace without clearance is increasing.

Infringements of controlled airspace by unknown aircraft present a significant risk to NATS. The safety need is to detect and bring to the controllers' attention infringements by unknown aircraft into controlled airspace. At present the system relies on controllers noticing unknown aircraft entering controlled airspace when monitoring the radar display. For aircraft that transpond a Secondary Surveillance Radar (SSR) code, the Short Term Conflict Alert (STCA) system provides a collision avoidance warning to the controller. However, because of the setting of the STCA parameters, separation may have already been lost and the airborne Traffic Alert and Collision Avoidance System (TCAS) may have already prompted the pilot to respond.

CAIT will be a new safety net tool to provide controllers with more timely warnings of controlled airspace infringements by aircraft. The intention is that CAIT will provide a solution to the safety need by improving the situational awareness for controllers, thus providing more time to plan actions to avoid a potential loss of separation and minimize the effects of the infringement.

During the CAIT project the authors worked with NATS domain experts to produce 1 SD model and 1 SR model of key actors, goals, soft goals, tasks and resources in airspace infringement detection using the extended versions of \(i^*\) and REDEPEND. This work took place independently of the CAIT software requirements specification, so as not to change the established requirements process in NATS. The \(i^*\) models and satisfaction arguments were produced and validated in 6 half-day meetings over a 4-month period. During each meeting 1 analyst facilitated the development and/or validation of parts of the models, whilst a second edited the models directly using REDEPEND. Development of these models was supplemented by a 1-day observation of work at the London Terminal Control Centre. The CAIT SD model was developed during the first and second meetings. Development of the SR model started in the second meeting and continued during the subsequent 4 meetings. The final version of the SR model is shown in Figure 1.
The model reflected the scale of the airspace infringement problem. It specified 25 actors, 15 of which were expanded with a total of 197 model elements – 77 tasks, 22 goals, 37 soft goals and 61 resources. These 197 elements were linked by a total of 299 links – 59 dependencies, 48 contribute-to soft goal, 73 means-end and 119 task decomposition links. Due to its size, we cannot provide a single readable version of the model in this paper (a readable version is available at [17]), so the next paragraphs highlight important elements. To develop the model we used REDEPEND features reported in [19] for zooming, filtering and highlighting different parts of the SD and SR models to the NATS domain experts. No major usability problems with the tool were reported.

The actor representing the new CAIT software system, on which requirements were specified in text form, is shown (A) in the model in Figure 1. The software system is a relatively small part of infringement detection. It shall undertake tasks such as produce alerts and record alerts using resources such as active alerting regions, to achieve soft goals such as false intruder alerts minimized. To do this it depends on resources that are produced by other actors, for example multi-radar track information produced by the Radar Data Processor (RDP) (B). The Controller Working Position (CWP) actor (C), which displays information to civil air traffic control officers (ATCOs), depends on the CAIT software system for the resource CAIT alerts (C), and CAIT achieving the soft goal alert production timely contributes positively to the CWP actor achieving the soft goal timely alerting, which in turn contributes positively to the civil ATCO achieving its soft goal timely detection.

The left-hand side of the model shows other actors that compose the NATS integrated air traffic control system, including aircraft that transmit Secondary Surveillance Radar (SSR) data, ground-based surveillance systems (D) that send data about aircraft positions and code, the RDP (B) that computes the accurate locations of aircraft from the data, and the Short-Term Conflict Alert (STCA) software system. The actor with the largest number of model elements, in the centre of the diagram (E), was developed to describe the tasks, resources, goals and soft goals of civil ATCOs using expert input during the meetings and models of cognitive controller behaviour [6]. The right-hand side of the model depicts pilots of both controlled (F) and uncontrolled (G) aircraft depending on the civil ATCO and each other to avoid collisions.

Although the NATS domain experts claimed that the SD and SR models were valuable, the models still suffered from poor specification of means-end links, and NATS lacked procedures to exploit the models during the analysis of the CAIT requirements. The next sections report the 2 extensions to i* in detail, then demonstrate them using CAIT satisfaction arguments developed by the NATS domain experts once the SR model was complete.

4. Integrated i*-Satisfaction Arguments

To extend i* semantics with satisfaction arguments we developed the conceptual model shown in Figure 2 to link concepts of i* means-end links and satisfaction arguments. We use the term means-end link to define both a means-end link which has a goal as the end and its specialization, a contributes-to soft goal link which has a soft goal as the end.

Figure 1: The i* SR model of the CAIT system, developed using the new version of REDEPEND
4.1 Model Concepts

An actor seeks to achieve or attain an end-element, which in i* can be a soft goal or a goal. An actor also has the means to achieve or attain the end-element. In i* a means can be a goal, a task, a resource or a soft goal (the latter 2 only being valid for a soft goal end). The actor seeks to attain a goal (a means to attain something else) and undertake a task (so that a goal might be attained). With soft goal contributes-to links, the achievement of one soft goal can contribute positively or negatively to achieving the other soft goal. The means-element and end-element are related using the i* means-end relationship. Where the end element is a soft goal, the relationship is attributed with values that specify the modality and type of contribution (Some+, Some-, Help, Hurt, Make, Break, Unknown) reported in [15] and supported in the RESCUE version of i*. This is depicted on the right-hand side of Figure 2.

Figure 2. Conceptual model that relates concepts from i* and satisfaction arguments

Each satisfaction argument is developed for one and only one end-element of a means-end link. The argument is constructed using one or more properties of the domain, one or more means-elements linked to the end-element, one attribute that explains the argument [4], and a second that expresses under which conditions the argument is applied to instances of the end-element. This is depicted on the left-hand side of Figure 2.

Satisfaction arguments require clear distinctions between phenomena visible to the system being designed and phenomena that are visible to the environment of the system. However in the CAIT project we needed to specify requirements on more than just the system to be developed. This was an important distinction from Jackson’s original conception of a satisfaction argument [18] and closer to the use of satisfaction arguments in REVEAL [4].

4.2 A Simple Example

Consider the 2 means-end links taken from the CAIT system SR model shown in Figure 3.

Figure 3. Two examples of means-end links taken from the CAIT system SR model

The first specifies that the achievement of the civil ATCO soft goal be aware of [infringement] alerts (the end-element of current human work required to change due to requirements on the new system) is contributed to positively by the task display picture (the means-element of a current system also required to change due to requirements on CAIT), which is undertaken by the CWP (the controller working position). The double-headed arrow specifies a MAKE contributes-to soft goal link – the contribution of the completed display picture task is positive and sufficient to satisfy the be aware of alerts soft goal. The second specifies that the achievement of the civil ATCO soft goal confidence in tools improved (the end-element) is contributed to positively by the actor’s own soft goal be aware of alerts (the means-element). The solid arrowhead specifies a HELP contributes-to soft goal link – the contribution of the be aware of alerts soft goal is positive and but not sufficient to satisfy the con-
Figure 4 shows a simplified satisfaction argument for the means-end link shown on the left-hand side of Figure 3. The end-element is be aware of alerts and the means-element is display picture. The operator always specifies that the argument is true in all instances. Properties of the domain that must be true if the CWP displaying picture shall contribute positively towards satisfaction of the requirement civil ATCO being aware of alerts are the ATCO is monitoring the air picture, and alerts are only communicated through the air picture. The explanation links the domain properties and means-element to the end-element. The argument is important for the introduction of CAIT even through CAIT does not form any element of the satisfaction argument, and the domain properties that must be true for CAIT to attain and achieve end-elements are different to the domain properties reported in Figure 4.

<table>
<thead>
<tr>
<th>Satisfaction argument</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End-element:</strong> Be aware of alerts</td>
</tr>
<tr>
<td><strong>Means-element:</strong> Display picture</td>
</tr>
<tr>
<td><strong>Condition:</strong> At all times</td>
</tr>
<tr>
<td><strong>Domain properties:</strong></td>
</tr>
<tr>
<td>- The ATCO is monitoring the air picture</td>
</tr>
<tr>
<td>- Alerts are only communicated through the air picture</td>
</tr>
<tr>
<td><strong>Explanation:</strong> The alert is reported with the air picture that is displayed with the CWP, to provide a single information source. It is the only source of alert information for the civil ATCO.</td>
</tr>
</tbody>
</table>

Figure 4. An example of a simplified satisfaction argument taken from the CAIT system model

This satisfaction argument is, we conjecture, more complete than the i* means-end link. In the next section we report the new procedure that uses i* models extended with such arguments to explore the impact of software requirements on safety-related goals and soft goals.

5. Exploring the Impact of Software Requirements on System-Wide Goals

We developed a procedure to explore the impact of software requirements on system-wide goals. The procedure uses i* SR models as reference models from which to infer candidate impacts of functional requirements on new software systems. It has 2 major stages:

1. Determine the impact of requirement compliance or otherwise on the successful completion of task elements and availability of resource elements of the new software actor in the SR model;
2. Determine the impact of these completed and uncompleted tasks and available and unavailable resources on the attainment of goals and achievement of soft goals in the SR model.

The procedure maps functional rather than non-functional requirements because of their predominance in NATS software requirements. Both stages are described.

5.1 Completing the Functional Requirement – SR Model Matrix

Analysts map functional requirements to SR model tasks and resources using a matrix, an example of which is depicted in Figure 5. Adding a value to a cell indicates that the requirement impacts on the task or resource. The value of the cell, a simple + or −, indicates whether the task or resource is enhanced or detracted. We sought to keep analyst completion by the analyst simple to ensure its uptake. For example, in Figure 5, the requirement CAIT alerts shall be logged automatically by the system will, if satisfied, enhance the successful completion of the task record alerts but will detract from the availability of the civil ATCO resource long-term memory of infringements. Similarly the requirement it shall be possible to set active regions for the CAS infringement alerts will, if satisfied, enhance the successful completion of the engineering task set active regions for CAIT.

![Figure 5. A part of a functional requirement – SR model matrix. NB: requirements are not original CAIT requirements for confidentiality reasons](image-url)

Analysts complete the matrix manually. We explored techniques to automate at least part of the procedure, for example mapping terms in each functional requirement to terms describing each task and resource. However the terseness of expressions and lack of context to determine word senses made such automation unreliable.

A completed matrix enables analysts to determine tasks and resources that have the potential to be non-compliant. A non-compliant element is any soft goal, goal, task or resource that, given the current software requirements, has the potential not to be achieved, attained, completed or made available. Analysts detect potentially non-compliant tasks and resources using simple patterns in the completion of the matrix:

1. **Requirements omission:** a task or resource that is not enhanced by any requirement can indicate missing requirements. Therefore analysts will need to determine the degree and nature of the impact of each non-compliant task and resource arising from requirements omission on system-wide goals and soft goals;
2. **Requirements detraction:** satisfying requirements can have unforeseen consequences that can make tasks and resources non-compliant. Again, analysts will need to determine the degree and nature of the impact of each non-compliant task and resource on system-wide goals and soft goals;
3. **Weak requirements compliance:** there are insufficient requirements to complete tasks or make resources available in all situations, thus making the tasks and
5.2 Determining Goal and Soft Goal Impacts

When the matrix is complete analysts use the procedure to determine if the impact of non-compliant tasks and resources causes goals and soft goals to become non-compliant. The procedure applies 6 propagation heuristics to all 4 types of i* model link reported in section 2. At the start of the procedure each non-compliant task and resource in the matrix is a non-compliant element:

\text{Continue until set\{non-compliant elements\} = empty.}

\text{For each non-compliant element:}
\text{1. IF dependee element in dependency relationship with depender element is non-compliant}
\text{THEN add dependee element to set\{non-compliant elements\}}
\text{AND go to next non-compliant element}
\text{ELSE consider next model element}

The 6 propagation heuristics use i* semantics to determine whether each linked model element can become non-compliant. The first 2 heuristics are deterministic and can be computed using tool support:

1. IF dependee element in dependency relationship with depender element is non-compliant
   THEN add dependee element to set\{non-compliant elements\}
2. IF task element decomposes into sub-task element
   AND at least 1 sub-task element is non-compliant
   THEN add means-element to set\{non-compliant elements\}

In contrast determining whether goals and soft goals are end-elements of means-end links is not deterministic, so the other 4 heuristics identify potential to become non-compliant. Deciding if the contributed-to element becomes non-compliant is where satisfaction arguments come in. The introduction of the software system as specified by the requirements might change domain properties that, if no longer true, invalidate the argumentation and make goal or soft goal non-compliant:

3. IF model element = end-element of means-end link
   AND at least 1 domain property in argument for means-end link = invalid
   THEN add means-element of means-end link to set\{non-compliant elements\}
4. IF model element = end-element of means-end link
   AND argument for means-end link = invalid
   THEN add means-element of means-end link to set\{non-compliant elements\}
5. IF model element = end-element of contributes-to soft goal link
   AND at least 1 domain property in argument for contributes-to soft goal link = invalid
   THEN add means-element of contributes-to soft goal link to set\{non-compliant elements\}
6. IF model element = end-element of contributes-to soft goal link
   AND argument for contributes-to soft goal link = invalid
   THEN add means-element of contributes-to soft goal link to set\{non-compliant elements\}

If we take the CWP task prioritise alerts, for example, and assume that the matrix has indicated it has become non-compliant, then we can explore whether its associated soft goals are also non-compliant. In this case, the CWP soft goal timely alerting is a potential non-compliant element, along with the Civil ATCO soft goal timely detection. Analysts can use the propagation heuristics to determine whether these soft goals are now non-compliant. For example, the Civil ATCO has an assumption associated with the task prioritise alerts that other detection measures are timely enough. Will this assumption still hold true when the new CAIT alert system is introduced?

6. Supporting i* Models, Satisfaction Arguments and Impact Analysis with a Tool

We implemented a new version of REDEPEND to support the specification of satisfaction arguments for i* means-end links and implement the procedure to analyze the impact of software requirements on system-wide goals and soft goals.

6.1 Specifying Satisfaction Arguments

To generate a new satisfaction argument an analyst selects a goal or soft goal in the SR diagram. REDEPEND automatically generates a new satisfaction argument sheet for the selected goal or soft goal using elements and links in the model as shown in Figure 6. The selected goal or soft goal is the default end-element, and each element that is a means to the goal or soft goal is a means-element. Means-elements are documented using 2 tabs. The internal tab displays means-elements from within the same actor boundary as the end-element, and the external tab displays means-elements from other actors.

The analyst manually specifies the conditions under which the satisfaction argument hold, and can change the end-element, means-elements and link types directly in the form. Because REDEPEND generates each satisfaction argument automatically from the SR model, such changes made by the analyst to the satisfaction argument sheet and model are propagated automatically to both, thus keeping each model and its arguments consistent.

The analyst manually completes each satisfaction argument using the domain properties section at the bottom of Figure 6. Domain properties are stored in a database of all domain properties associated with an SR model to ensure effective reuse of properties that, we believe, can improve the specification of satisfaction arguments. For each domain property in an argument, the analyst selects
6.2 Completing the Functional Requirement-SR Model Matrix

An analyst completes each functional requirement-SR matrix using a spreadsheet embedded in REDEPEND. Part of the matrix developed for CAIT is shown in Figure 8. An analyst copies functional requirements into the left column, then REDEPEND automatically generates the other columns with tasks and resources from the selected software actor in the SR model – in this case the CAIT actor. The analyst then completes the matrix. To aid this task REDEPEND supports 2-way navigation between elements in the SR model and the matrix. If an analyst selects a matrix column, then REDEPEND will highlight the corresponding element in the SR diagram. Likewise if an analyst selects an element in the SR model the matrix will reposition to the corresponding column. Figure 8 demonstrates how an analyst can toggle quickly between a selected matrix row and the corresponding SR model element (record alerts task) on the diagram depicted on Figure 1. We consider such model navigation is essential to support the analysis of large systems such as CAIT.

6.3 Supporting Impact Analysis

REDEPEND also has new features to support SR model walkthroughs for determining propagation impacts on goals and soft goals. The analyst can tag an SR model element as non-compliant, and then use REDEPEND to detect and tag other elements that are potentially non-compliant by using propagation heuristics to follow the i* links upwards – means to end, depends to dependee and task-decomposition to composite task. Non-compliant elements are depicted with an X and undecided elements with a ?.

Figure 9 shows the results of one walkthrough. The monitor air picture task is tagged as non-compliant whilst the soft goal timely detection is undecided. The analyst is using REDEPEND to apply automatically the propagation heuristics to the CAIT SR model. Whilst propagation across dependency and task-decomposition links is automatic, analyst judgment is needed to decide whether propagation happens across means-end and contributes-to soft goal links based upon detection of untrue domain properties and invalidated explanations using the satisfaction argument sheets.

7. Using Satisfaction Arguments in CAIT

NATS domain and requirements analysts applied satisfaction arguments and the new procedure to CAIT. The domain analysts developed 8 satisfaction arguments for selected means-end links in the SR model. Each argument contained, on average, 1 specification means, 2 refinement means and 3 domain properties. The database associated with the SR model contained a total of 25 domain
properties, suggesting little reuse of CAIT domain properties during development of the satisfaction arguments. Nonetheless, NATS analysts were able to document satisfaction arguments.

Figure 9. Explore propagation impacts in REDEPEND

Development of the i* models and satisfaction arguments invited favorable comparisons with other goal-based techniques such as KAOS [3,14]. Whilst the temporal logic constructs of KAOS are well suited to provide arguments for requirements satisfaction for concrete goals, this is often at the cost of complexity and usability. The analysts concluded that REDEPEND provided a more lightweight approach to record reasoning about goals and soft goals that, in socio-technical systems cannot be formally proven to be satisfied. Establishing boundaries of the CAIT socio-technical system was a critical task. Whilst KAOS can distinguish between goals as requirements and assumptions, it lacks a comprehensive equivalent of a context diagram. The analysts reported that the i* SD model functioned well in this regard.

Once 2 i* models and satisfaction arguments were available, two requirements analysts – one from the CAIT team – applied the procedure to analyze the impact of 26 CAIT software requirements on safety-related goals. The reporting of this evaluation is beyond the scope of this paper. We hope to report the results in the near future.

8. Research Contributions Revisited

As reported in section 2 we claim 4 novel contributions for the research. The first is the conceptual integration of i* and satisfaction arguments. Although satisfaction arguments in requirements projects are not new, we argue that their integration with models such as i* is. REVEAL represented satisfaction arguments in simple tables [4] that are not tightly integrated with modelling techniques. Our integration of satisfaction arguments with i* provides a mechanism with which to improve the completeness of such models by technique combination.

The second claim – for a new impact analysis procedure – is partially supported. Our impact analysis procedure is similar to Horkoff et al.’s i* evaluation procedure [5]. Both use i* semantics to propagate values of elements through an i* model. However, our use of integrated satisfaction arguments is, we believe, novel. Horkoff’s procedure uses human judgment based on unspecified contextual knowledge to determine propagations. In contrast our impact analysis procedure uses satisfaction arguments to provide a scaffold with which to document contextual knowledge and guide human reasoning about it. Moreover satisfaction arguments can increase the completeness of i* models with more problem domain knowledge without overloading the i* notation, which was a cited reason not to formalize documentation of contextual knowledge in the i* evaluation procedure [5].

This leads to our third claim – novel software features for i* modeling and analysis. To support the reported impact analysis procedure REDEPEND provides analysts with tightly integrated development of i* models and specification of satisfaction arguments. Whilst its automated propagation heuristics are a simplified subset of the propagation rules reported in [5], the REDEPEND integrated toolset delivers new capabilities to human analysts to explore the impact of software requirements on system-wide goals satisfaction arguments in a coherent manner.

Our fourth claim is that the integrated REDEPEND toolset and procedure scale to large problems. The CAIT SR model has over 400 elements and links – the largest i* model developed by us so far. So we claim preliminary evidence for scalability. Whilst NATS domain analysts expressed concern about the time needed to develop and validate the i* models, this was due to the complexity of the problem domain rather than limitations with REDEPEND. The upfront effort needed to generate large i* models suggests that such models have a future role as reference models reused in multiple projects to analyze the impact of software requirements. We have yet to explore the potential of reusing the CAIT i* models in other NATS and air traffic management projects.

Determining whether the integrated REDEPEND tool and procedure was useful in CAIT is the subject of ongoing analysis not reported in this paper. Preliminary evidence is encouraging. NATS analysts documented satisfaction arguments that added domain properties and argumentation to i* means-end links that would have been otherwise missing from the model.

9. Conclusions and Future Work

This paper reports work reported in an earlier form in [20]. Although our integration and empirical evaluation of i* modeling and satisfaction arguments was preliminary
we are able to draw initial conclusions that will inform future research in this direction.

The i* SD and SR models produced during CAIT and the satisfaction arguments developed from the models indicate that experienced analysts were able to extend means-end links with 8 satisfaction arguments to support the analysis of a complex socio-technical system. The analysts were able to develop and reason using satisfaction arguments linked to means-end links in the CAIT i* SR model. The satisfaction arguments expressed domain properties that were not explicitly represented with i* semantics in the SR model, thus adding to the model’s completeness with respect to domain knowledge considered important by the analysts during the specification of the CAIT system. These preliminary successes provide some empirical evidence for our conceptual integration of i* modeling and satisfaction arguments.

The successes also provide directions for future work. Having demonstrated the potential of the integration we will improve the robustness of REDEPEND then apply it to model and analyze other socio-technical systems. The main difference will be to develop satisfaction arguments alongside the i* SR model, to investigate satisfaction arguments as a technique to structure and document discussions, similar to use of argumentation tools such as [1]. We will also investigate whether reasoning about system boundaries in terms of phenomena visible to actors and their environments in the form of satisfaction arguments can positively inform the development of SR models.

Results from the use of the procedure to investigate the impact of CAIT system requirements on safety-related goals and soft goals are pending. Once available we will use the results to improve the procedure and support provided by REDEPEND for it.

Acknowledgements
The research reported in this paper was supported by NATS, the UK national air traffic service.

References