Exploring the Impact of Software Requirements on System-Wide Goals: a Method using Satisfaction Arguments and *i** Goal Modelling

James Lockerbie [1], Neil Maiden[1], Jorgen Engmann[2], Debbie Randall[3], Sean Jones [3], David Bush [3]

City University London[1]
Centre for Human-Computer Interaction Design
London EC1V 0HB, UK
James.Lockerbie.1@city.ac.uk, +44 (0)20 7040 8326
N.A.M.Maiden@city.ac.uk

Health Protection Agency[2]
Centre for Infections
61 Colindale Avenue
London, NW9 5EQ, UK
rmjdjel@ucl.ac.uk

NATS Corporate & Technical Centre[3]
4000 Parkway, Whiteley
Fareham, Hants PO15 7FL, UK
Debbie.Randall@nats.co.uk, Sean.Jones@nats.co.uk, David.Bush@nats.co.uk

Abstract

This paper describes the application of requirements engineering concepts to support the analysis of the impact of new software systems on system-wide goals. Requirements on a new or revised software component of a socio-technical system not only have implications on the goals of the subsystem itself, but they also impact upon the goals of the existing integrated system. In industries such as air traffic management and healthcare, impacts need to be identified and demonstrated in order to assess concerns such as risk, safety, and accuracy.

A method called PiLGRIM was developed which integrates means-end relationships within goal modelling with knowledge associated with the application domain. The relationship between domain knowledge and requirements, as described in a satisfaction argument, adds traceability rationale to help determine the impacts of new requirements across a network of heterogeneous actors. We report procedures that human analysts follow to use the concepts of satisfaction arguments in a software tool for i goal modelling. Results were demonstrated using models and arguments developed in two case studies, each featuring a distinct socio-technical system – a new controlled airspace infringement detection tool for NATS (the UK's air navigation service provider), and a new version of the UK's HIV/AIDS patient reporting system. Results provided evidence towards our claims that the conceptual integration of i* and satisfaction arguments is usable and useful to human analysts, and that the PiLGRIM impact analysis procedures and tool support are effective and scalable to model and analyse large and complex socio-technical systems.*

Keywords

*i** modelling, satisfaction argument, impact analysis, requirements process

1.Introduction

The ability of analysts to analyse change requirements in socio-technical systems has become increasingly important as projects and systems increase in size and complexity. There is a need to be able to demonstrate the impacts of new system requirements on existing and future system-wide goals, so that concerns such as safety, risk and accuracy can be assessed. In order to do this, we sought to develop a useful and usable method for analysts to follow that integrates research-based requirements techniques successfully into established industrial requirements processes.

Analysts are increasingly using *i**, the strategic goal modelling approach [37], to model and analyse requirements on socio-technical systems. *i** has been applied successfully to model requirements for air traffic management tools [22, 23] and decision support aids in agriculture [31] as well as to support individuals and groups in the work of charitable organizations [33]. Reported benefits to these projects have included automatic requirements generation from *i** models [23] and detection of omissions from UML requirements specifications [22]. However, our experiences with *i** in these projects also revealed two important weaknesses: (i) inadequate semantics to express the relationship between a means specification and an end goal, and: (ii) poor integration with in-house requirements processes. In this paper we report results from research that extended *i** to overcome these two weaknesses.

The first research extension was to add concepts from rich traceability [3] and satisfaction arguments [9] to provide additional semantics for modelling means-end relationships. The second was to develop new manual procedures with which to exploit *i** models more effectively in requirements processes. These procedures, and the tools developed to support analysts to follow the procedures, were designed to explore the impact of new software requirements on system-wide goals such as safety and accuracy. Proof of concept of both extensions was demonstrated with two requirements project case studies.

The first study was 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 by analysts to explore the impact of the new tool requirements on the safety-related goals of the wider air traffic management system. The second study was undertaken at the UK's Health Protection Agency (HPA). An analyst applied the same methods used in the first study to explore the wider impacts of implementing change request requirements on an HIV/AIDS Patient (HAP) reporting system.

We claim five 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 so that analysts can analyse the impact of specified software systems on system-wide goals. The third is a set of novel software features that enable analysts to extend SR models with satisfaction arguments, and that support and exploit the new procedure. All are demonstrated using models and requirements developed to address complex requirements engineering problems in the areas of air traffic management and patient surveillance. Hence, our fourth claim is that our techniques and their support software tools scale to model large and complex socio-technical systems with *i**. Finally, our fifth claim is that our methods are useable and useful to requirements analysts.

The remainder of the paper is in 9 sections. Section 2 describes the two weaknesses with *i** uncovered in previous projects, and outlines our solutions to overcome these

weaknesses. Section 3 presents the method, called PiLGRIM, in detail and Section 4 reports how the method was implemented in our *i** modelling tool. Section 5 presents the first case study, the airspace infringement detection problem to which NATS applied our solutions. Section 6 presents the second case study, where an HPA analyst used our approach to investigate the impact of requirements changes on system-wide goals in the area of patient surveillance. Sections 7 and 8 summarize related work and review claims made for the reported research, then Section 9 outlines potential threats to its validity. The paper ends with our plans for future development of PiLGRIM and the REDEPEND tool.

2. Previous Work

2.1 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 [37]. The strategic actor is the central concept of *i**, and is viewed as having intentional goals, beliefs, abilities and commitments. The intentional aspects associated with any actor can be characterised by four process elements: (i) a *goal* represents a condition or state of the world that can be achieved or not, but does not describe how it can be achieved; (ii) a *task* represents one particular way of attaining a goal, and can therefore be considered as a detailed description of how to accomplish a goal; (iii) a *resource* can be considered as an entity used in, or the product of, some process or a task; and (iv) a *soft goal* relates to the notion of a goal that cannot be so sharply defined, such as goals that describe properties or constraints of the system being modelled. *i** is an established approach for goal modelling, and has given rise to different versions of syntax and semantics that support different styles and uses of *i** modelling. Most versions support the development of two 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 wants, 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 accomplish something by themselves or depend on other actors. The SR model has four main types of element: *goals*, *tasks*, *resources* and *soft goals*. These four types can be linked using the four relationship links available within the SR modelling semantics: 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** modelling with a software tool called REDEPEND [21], which extends Microsoft Visio with features specific to *i** to enable requirements analysts to model and analyse 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. Features include automatic checking of *i** syntax, change synchronisation features between linked

models, and check features to highlight and shade-out model elements for partitioning and marking up models during analysis and review tasks. Indeed, both direct manipulation and automated model verification are seen as essential for scalable *i** modelling and tool-supported analysis procedures.

In the last 6 years we have applied *i** and REDEPEND to model requirements for four major air traffic management systems, including a departure management system for major European airports [22] and a system that supports the scheduling of UK airspace [25]. Whilst *i** provided important new capabilities in these projects that we have reported elsewhere [23], the projects also revealed the two weaknesses that, we argue, need to be overcome to ensure widespread industrial uptake of *i**.

2.2 Encountered Weaknesses with *i**

The first encountered weakness is the *i** means-end link. In *i** a task specifies one way in which to achieve a desired state, and can be decomposed into sub-components by the task decomposition link. The means-end link is used to associate tasks, and their decompositions, with states that actors want to achieve or attain – goals and soft goals. The means-end link can also reduce goals to sub-goals where the end is the top-level goal. However, analysts in our previous projects, when expressing means-end links, reasoned about much more than just the means elements 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.

Whilst *i** provides means-end contribution types (*Some+, Some-, Help, Hurt, Make, Break, Unknown* reported in [38]), these alone are not sufficient to explore impacts of system specifications on goal attainment. Although *i** includes the concept of a *belief*, a form of claim applied to means-end links to express actors' different beliefs [38], the argumentation behind the association of these beliefs is unclear. 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. Therefore we needed a method extension that would support previously observed analyst practices more effectively.

The second weakness is a lack of guidance for requirements analysts on how to embed research-based techniques such as *i** in requirements processes. For example, 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. Organisations such as NATS and the HPA cannot overturn their established requirements processes to accommodate new *i** modelling. Therefore we explored new ways of developing, documenting and using *i** models and their extensions in processes that express software requirements in text form. We further explored this approach in our second study, analysing how the established method of change request requirements on the HAP system impact on actors in the wider HIV/AIDS reporting system.

2.3 First Extension: Satisfaction Arguments for *i**

To address the first weakness, we sought to exploit established techniques for rich

traceability and satisfaction arguments from reported requirements methods to extend means-end link modelling in *i**. Hammond et al. [9] explored the anatomy of a satisfaction argument based on Jackson's [17] distinction between the machine and the world. Jackson recognized the role of domain knowledge in the requirements process, and suggested a relationship that the REVEAL requirements process labelled a satisfaction argument to enhance the rich traceability of the process. This relationship defines domain knowledge as properties of the world that are known or assumed to be true, and requirements as properties of the world that are wished to be true. This combination of domain knowledge and system knowledge (specification) is the basis of the satisfaction argument: Using the relevant properties of the application domain D, when combined with the specification of the machine S to be constructed, it is possible to show that the requirements will hold [9].

This approach has led to the recognition that many problems with requirements arise as a result of problems with domain knowledge rather than problems with the system specifications or requirements statements. Furthermore, even if domain knowledge is accurately captured such that initial requirements hold, changes in the domain could result in requirements not being met anymore [9]. Likewise, new requirements may lead to the invalidation of domain assumptions.

Our use of satisfaction arguments, based on REVEAL, also draws upon Dick's [3] use of traceability rationale to enhance understanding. In rich traceability a satisfaction argument normally applies to a set of links associated with a requirement, which translate to all means-end links associated with an end in a *i** model. Such satisfaction arguments can be applied for two purposes – *sufficiency* and *necessity*. According to Dick [3], sufficiency explores whether the set of design artefacts are sufficient to satisfy the customer requirement. Necessity explores whether each of the design artefacts are necessary to satisfy the customer requirement. When applied to *i** modelling, we can interpret sufficiency as exploring whether the means and associated domain properties are sufficient to lead to the end, and necessity as whether each of the means and associated domain properties are necessary to lead to the end.

2.4 Second Extension: Using *i** Models to Explore How Software Requirements Impact Important Organisational Goals

To address the second weakness, we sought to develop a new procedure that requirements analysts could follow to use *i** models in traditional requirements processes that express software requirements in text form.

Linking *i** models to text requirement specifications is not new. Indeed, in two previous air traffic management projects we generated candidate requirement statements from *i** models using a pattern-based approach [23]. However, one limitation was that the *i** models had to be generated first. This was not possible in many requirements processes such as those used in projects at NATS and the HPA, as requirements on a new or revised software component of the socio-technical system had already been expressed. As a result, there was a need for change management, and we proposed a process of rich traceability and impact analysis to address this. Dick [3] describes the benefits of such an approach as the increased understanding of the relationships between design layers, and the possibility for semi-automated impact analysis to assess the potential impact of change.

Therefore, we developed a new procedure that requirements analysts could follow to explore the impact of documented software requirements on system-wide goals using *i** models. Our aim was for an analyst to be able to assess the impact of the new

software system on the attainment and achievement of actor goals and soft goals that, in our two studies, were largely concerned with safety and accuracy respectively.

3. The PiLGRIM Method

*i** modelling, satisfaction arguments, impact procedures and the REDEPEND tool were integrated into a method called PiLGRIM (Propagating i*-Led Goal-Requirement Impacts). The overview presented in Figure 1 shows the logical flow of the four stages of the method. The method starts with goal modelling in stage 1. The first and second stages take place in parallel throughout the method as analysts add and revise satisfaction arguments within the extended SR model to provide structured support for previously observed analyst behaviour. Once the underlying SR model is finalised, the matrix completion and propagation stages are required to run sequentially in order to generate a complete set of results. Each stage's procedure is described in turn in the following sections.

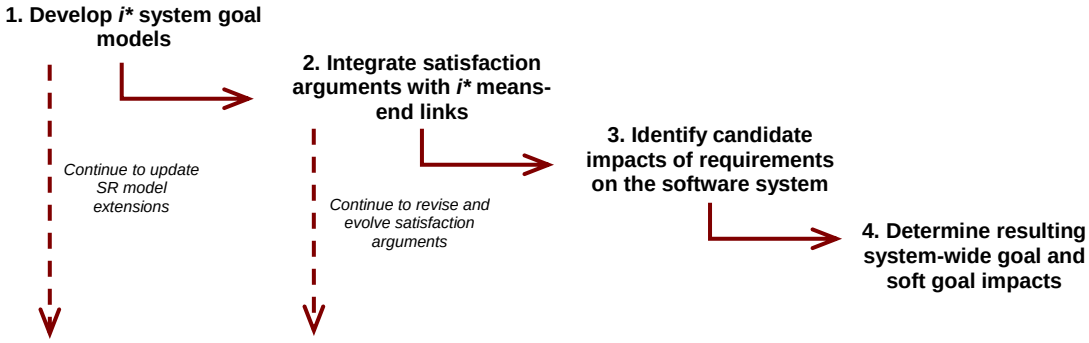


Figure 1: An overview of the 4 stages of the PiLGRIM method

3.1 Develop *i** System Goal Models

For the first stage of the PiLGRIM method we took our existing approach to *i** modelling using REDEPEND as described earlier in Section 2.1. This approach supports version 5.0 of *i** tailored to RESCUE, our requirements engineering process [19], based on recommendations to enhance the notation and tool features from analysts in previous projects. Whilst analysts were generally satisfied with the *i** syntax, they identified an absence of content and guidance to facilitate documentation and human reasoning with it – hence our method extension. Styles of modelling not supported in other *i** versions includes means-end links across actor boundaries which enable the modelling of complex trade-offs in the development of socio-technical systems. This modelling variation is used in the integrated *i**-satisfaction arguments described next.

3.2 Integrate Satisfaction Arguments with *i** Means-End Links

To extend *i** semantics with satisfaction arguments we developed a conceptual model to link concepts of *i** means-end links and satisfaction arguments defined across socio-technical system boundaries. For the PiLGRIM method we use the term *means-end link* to define both a means-end link which has a goal as the end and its specialisation, a contributes-to soft goal link which has a soft goal as the end.

3.2.1 Satisfaction Arguments for Socio-technical Systems

Satisfaction arguments distinguish between specified properties of a new system and assumed properties of the domain. Jackson [40] originally assumed a single system and boundary so that properties were attributed simply to the system or domain. However socio-technical systems have multiple systems and actors whose

work is redesigned, for example redesigning the work of an air traffic controller who is using new airspace infringement software. Therefore arguments for the satisfaction of requirements of these systems and actors need to be based on both properties assumed about the domain and properties of other systems – i.e. whether known properties of current systems hold and requirements on these new systems and actors are satisfied.

As a consequence the PiLGRIM Method needs to support analysts to specify different satisfaction arguments for different boundaries in a socio-technical system. Analysts can specify requirements on new software system(s) and other systems and human work to change as specified in requirements. The result is that an end-element of one satisfaction argument can fulfil the role of a means-element in another – a goal to be achieved by one system is assumed achieved by the other.

To enable this to happen, REDEPEND supports analysts to tag each actor in an *i** model as a new system, current actor or system required to change, or an actor outside of the project boundaries and therefore not within the redesign scope. This enables traceability between different system boundaries to be specified explicitly in a satisfaction argument. A means element contributing to an end element inside the same actor boundary is classified as a *refinement*, while a means contributing from outside the boundary is classified as a *specification*, and is qualified with the actor name. Figure 2 illustrates these features in a simple schema. The satisfaction argument specified in *Actor B*, an existing system to change, contains one *refinement* means, a *specification* means from the new system, *Actor A*, and a *specification* means from *Actor C* which is outside of the redesign scope of the project.

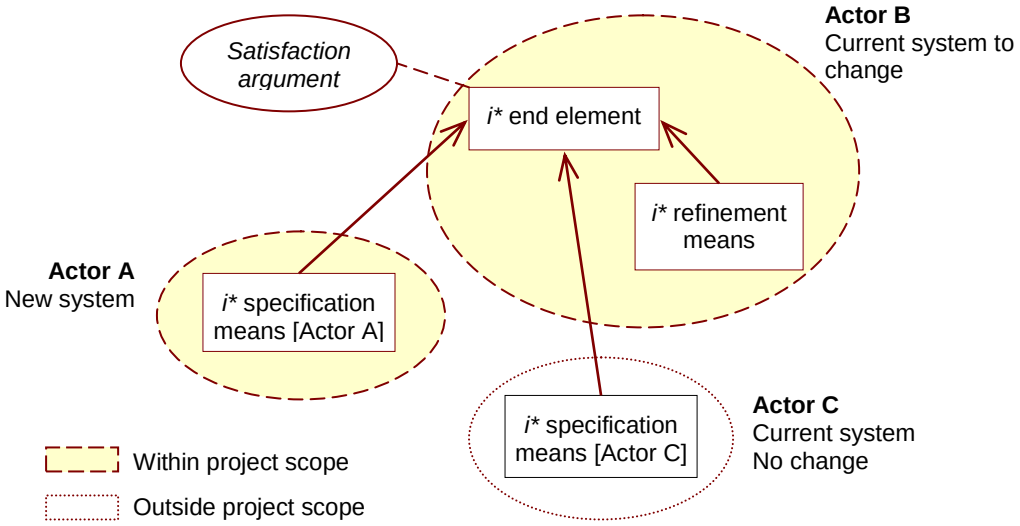


Figure 2: An example showing how the PiLGRIM method uses actor types and refinement and specification means in satisfaction arguments

3.2.2 Model Concepts

The PiLGRIM method is underpinned with a simple meta-model, depicted in Figure 3, that combines *i** and satisfaction argument 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 [38] and supported in the RESCUE version of *i**. This is depicted on the right-hand side of Figure 3.

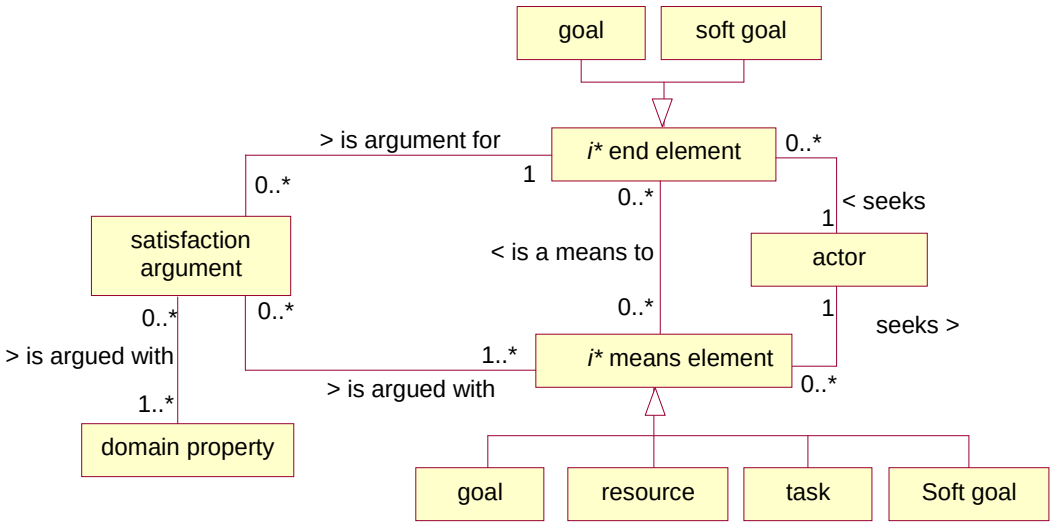


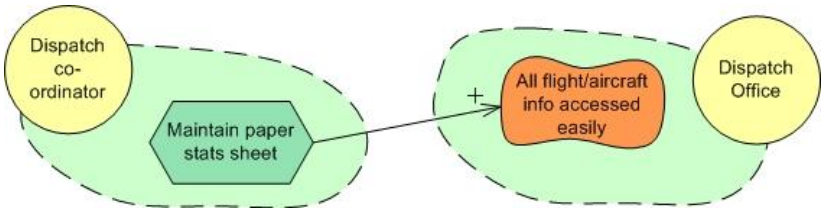
Figure 3: Conceptual model that relates concepts from *i and satisfaction arguments**

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, and one attribute that explains the argument [9]. This is depicted on the left-hand side of Figure 3. Analysts then apply the principles of *sufficiency* and *necessity* from rich traceability in order to determine whether the means and associated properties are sufficient to lead to the end, and whether each of the means and associated properties are necessary to lead to the end. This helps to produce a complete argument, ensuring that the end goal is met whilst also protecting against over-engineering the solution [3].

For example, in VANTAGE, an earlier air traffic management project [29], we produced an *i** SR model showing how enhanced airport operations could minimise the environmental impact of a regional airport. The model included a means-end link that showed that successful completion of the *dispatch co-ordinator* task *maintain paper stats sheet* contributed positively to the *dispatch office* achieving the soft goal *all flight/aircraft info accessed easily*. The stats sheet fulfilled an important role in the functioning of the dispatch office, as it contained information such as passenger figures and estimated aircraft arrival times. However, this representation in *i** alone did not contain enough information to justify the means being a means to the end, i.e. it was not possible to determine sufficiency and necessity without richer traceability of the relationship.

Figure 4 shows the means-end link and a simplified version of its accompanying satisfaction argument. The end-element is the soft goal *all flight/aircraft info accessed easily* and the means-element is *maintain paper stats sheet*, modelled as a task as it represents one particular way of attaining the soft goal (the other considered alternative being an electronic VANTAGE version). Three properties of the domain that must be true for the means to contribute to the end are: *the paper stats sheet is the*

single common repository of all information related to the aircraft/flight, dispatch office workers only seek info from the paper stats sheet, and *all staff follow established dispatch office work practices*. The explanation part forms the argument by linking the domain properties and means-element to the end-element.



Satisfaction argument

End-element: All flight/aircraft info accessed easily
Means-element: Maintain paper stats sheet
Domain properties:

- The paper stats sheet is the single common repository of all information related to the aircraft/flight
- Dispatch office workers only seek info from the paper stats sheet
- All staff follow established dispatch office work practices

Explanation:
The paper stats sheet is used to record data from a variety of sources, including telexes and airport management systems. It is, in essence, a one-stop shop for all aircraft/flight information required by the dispatch office

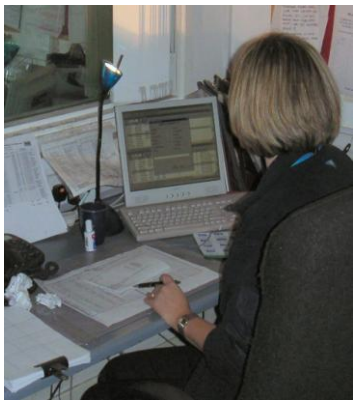


Figure 4: An example of a simplified satisfaction argument taken from the VANTAGE socio-technical system model

This satisfaction argument is, we conjecture, more complete than the original *i** means-end link. It has the potential to support more naturalistic analyst reasoning observed in previous requirements projects using *i**. Next, we report the new procedure that uses *i** models extended with such arguments to explore the impact of software requirements on system-wide goals.

3.3 Identify Candidate Impacts of Requirements on the Software System

The third stage of the PiLGRIM method uses the enhanced *i** SR models as reference models from which to guide analysts to infer candidate impacts on the socio-technical system from functional requirements on the new, or updated, software system. The procedure maps functional rather than non-functional requirements, not only due to their predominance in software requirements, but also because they represent features that we aim to explore the impact of. Analysts are guided to cross-check each non-functional requirement with the soft goals in the SR model to ensure consistency and completeness for assessing the impacted goals and soft goals.

To identify the impacts, analysts map functional requirements to SR model tasks and resources using a matrix, as requirements and functional *i** elements need not follow a one-to-one relationship. An example is depicted in Figure 5. Adding a simple + or – value to a cell indicates how the requirement, if satisfied, will impact on the task or resource. The tasks and resources are indicated to be either *compliant* (+) or *non-compliant* (–) with the functional requirements specification of the introduced, or

updated, software system.

To develop the matrix the requirements analysts were asked to consider each requirement in turn using the following method guidance:

1. *Compliant task:* If the requirement in the system specification, when satisfied by the new system, will enable the actor to complete the task successfully, compared with undertaking the task in the AS-IS system, then link the requirement to the task with a compliant (+) value.
2. *Non-compliant task:* If the requirement in the system specification, when satisfied by the new system, will detract from the actor's ability to complete the task successfully, compared with undertaking the task in the AS-IS system, then link the requirement to the task with a *non-compliant* (–) value.
3. *Compliant resource:* If the requirement in the system specification, when satisfied by the new system, will enable the actor to obtain the resource successfully, compared with obtaining the resource in the AS-IS system, then link the requirement to the resource with a *compliant* (+) value.
4. *Non-compliant resource:* If the requirement in the system specification, when satisfied by the new system, will detract from the actor's ability to obtain the resource successfully, compared with obtaining the resource in the AS-IS system, then link the requirement to the resource with a *non-compliant* (–) value.

Figure 5 presents a continuation of our example from the VANTAGE project. It shows that the requirement *the VANTAGE system shall manage the daily mayfly stats sheet* would, if satisfied, be compliant with the successful completion of the *airport management* task *optimise operational efficiency*. In contrast, the same requirement would, if satisfied, have been non-compliant with the successful completion of the current *dispatch co-ordinator* task *maintain paper stats sheet*.

Functional requirement	Airport Management: Optimise operational efficiency [Task]	Dispatch Co-ordinator: Maintain paper stats sheet [Task]
The VANTAGE system shall manage the daily mayfly stats sheet	+	–

Figure 5: Part of a functional requirement – SR model matrix for the VANTAGE system

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 the *i** model element expressions and lack of context to determine word senses rendered such automation unreliable.

Matrix completion uses simple patterns. The first 2 patterns relate to requirements that do not map to SR model tasks and resources:

1. *Unnecessary requirements:* requirements that do not impact any tasks or resources in the model can indicate unnecessary requirements. Therefore analysts will need to determine whether these requirements should be removed from the specification.
2. *Missing tasks and resources:* requirements that do not impact any tasks or resources in the model, but are deemed necessary by analysts e.g. because they

impact goals, can indicate missing tasks and resources. Therefore analysts will need to revise the SR model in line with the specification.

The final 3 patterns then enable the analysts to detect potentially *non-compliant* tasks and resources. 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:

3. *Requirements omission*: a task or resource that is not impacted 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;
4. *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;
5. *Weak requirements compliance:* there are insufficient requirements to complete tasks or make resources available in all situations, thus making the tasks and resources potentially *non-compliant*.

3.4 Determine the Resulting System-Wide Goal and Soft Goal Impacts

The final stage of the PiLGRIM method builds on the integrated *i**-satisfaction arguments and impacted tasks and resources to explore the attainment of goals and achievement of soft goals in the SR model.

When the matrix is complete analysts use the final procedure to determine if the impact of *non-compliant* tasks and resources causes goals and soft goals to become *non-compliant*. Whilst it is also possible to demonstrate compliance through the SR model, the method prioritises propagating non-compliance in order to make the most effective use of time and resources e.g. domain experts.

The procedure applies 6 propagation heuristics to all 4 types of *i** model links reported in section 2. At the start of the procedure each *non-compliant* task and resource in the matrix is a *non-compliant* element:

```
For each non-compliant element:
 For each model element linked to the non-compliant element:
  IF at least 1 propagation heuristic applies to model element
  THEN add model element to set{non-compliant elements}
  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, concerning propagation across dependency relationships and task decompositions, are deterministic and non-compliance can be computed automatically using tool support:

```
1.  IF model element = depender element in dependency relationship
    AND is non-compliant
    THEN add dependee element to set{non-compliant elements}

2.  IF model element = decomposition of a task
    AND is non-compliant
    THEN add task element to set{non-compliant elements}
```

For propagation across means-end and contributes-to links, non-compliance is not

computationally deterministic. An end element with one or more non-compliant means is automatically identified as having the potential to become *non-compliant*. Deciding if the contributed-to element becomes *non-compliant* is where the satisfaction arguments come in through the application of the next 4 heuristics. 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 = invalid
   THEN add end-element of means-end link to set{*non-compliant elements*}
4. IF model element = end-element of contributes-to soft goal link
   AND at least 1 domain property in argument for contributes-to soft goal = invalid
   THEN add end-element of contributes-to soft goal link to set{*non-compliant elements*}

If analysts cannot rewrite the argument so that the goal is attained or soft goal is achieved, then the goal or soft goal is *non-compliant*. On the other hand the introduction of the software system that meets the specified requirements might not change domain properties but it might have specifications that invalidate the argument and make the goal or soft goal *non-compliant*:

5. IF model element = end-element of means-end link
   AND argument for means-end = invalid
   THEN add end-element of means-end 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 = invalid
   THEN add end-element of contributes-to soft goal link to set{*non-compliant elements*}

Again if the analysts cannot rewrite the argument so that the goal is attained or soft goal achieved, then the goal or soft goal is *non-compliant*.

Returning to our earlier example, we took the non-compliant *dispatch co-ordinator* task *maintain paper stats sheet* and explored whether its associated soft goals in the VANTAGE *i** SR model were also *non-compliant*. Application of the heuristics revealed that *dispatch office* soft goals related to the affordances of the paper version of the stats sheet were potential *non-compliant* elements after the introduction of the new computerised VANTAGE system. We then used the propagation heuristics to determine whether these soft goals were now indeed *non-compliant*. Figure 6 demonstrates the process undertaken by a human analyst with automated support from the REDEPEND software tool for one of these soft goals – *all flight/aircraft info accessed easily*. Analysts needed to consider whether the domain assumptions would still hold true following the introduction of the new VANTAGE system. As the figure shows, the final assessment determined that all three domain properties were in fact invalidated by the new specification, and therefore the soft goal was deemed to be *non-compliant*.

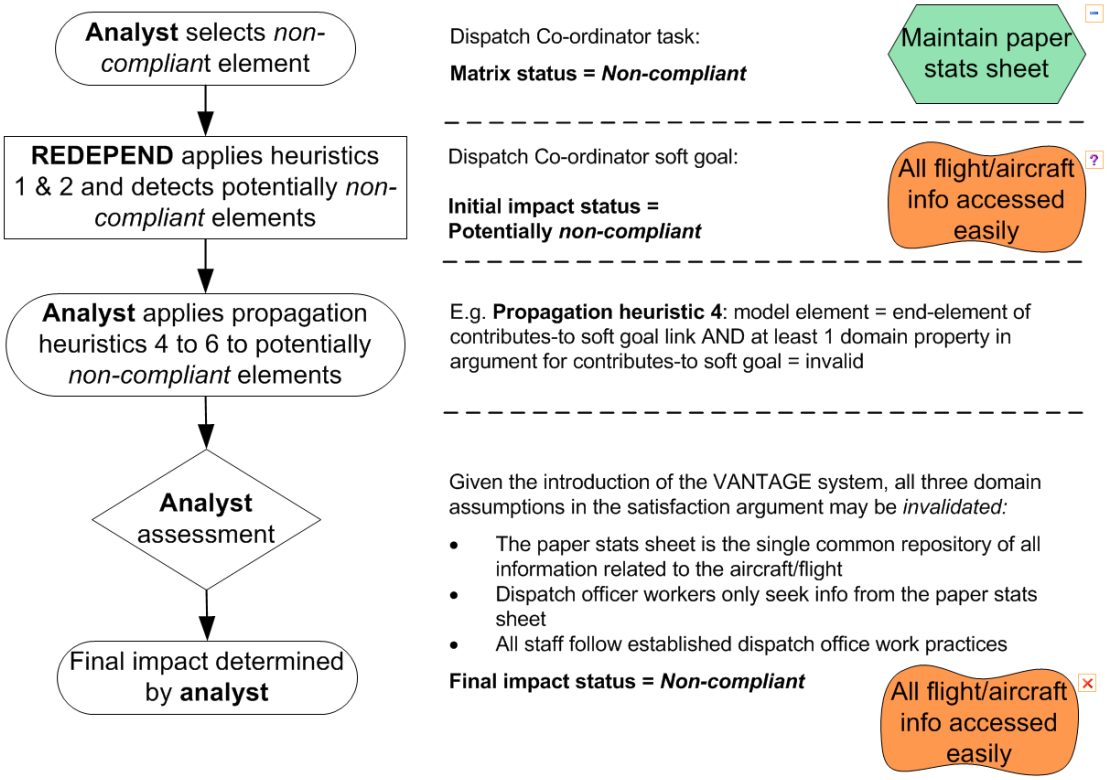


Figure 6: An example application of the propagations heuristics to the *maintain paper stats sheet* task from the VANTAGE system model

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

We implemented a new version of REDEPEND to support requirements analysts to specify satisfaction arguments for *i** means-end links and analyse the impact of software requirements on system-wide goals and soft goals using the procedure. The tool provides automatic model analysis functions to help the analyst undertake human reasoning.

4.1 Specifying Satisfaction Arguments

To generate a new satisfaction argument an analyst selects a goal or soft goal in the *i** SR diagram. REDEPEND automatically generates a new satisfaction argument sheet for the selected goal or soft goal using elements and links in the model. Figure 7 shows the REDEPEND representation of the satisfaction argument for the soft goal *all flight/aircraft info accessed easily* presented earlier in Figure 4. As can be seen, 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 refinement means-elements from within the same actor boundary as the end-element, and the *external* tab displays specification means-elements from other actors.

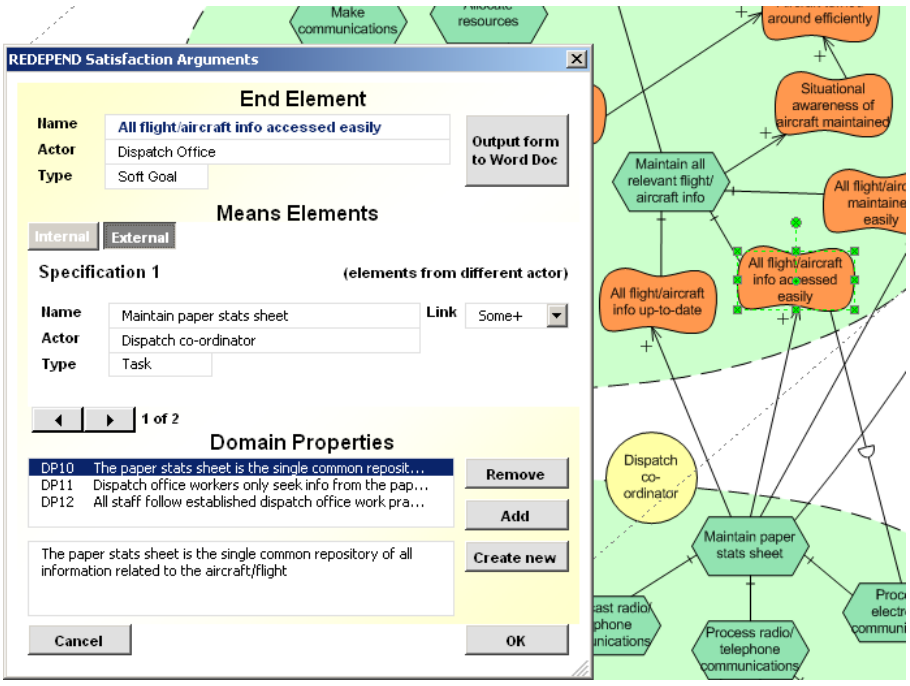


Figure 7: One satisfaction argument showing the *maintain paper stats sheet* specification taken from the VANTAGE model, specified in REDEPEND

The analyst can change the 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 the model and its arguments consistent.

The analyst manually completes each satisfaction argument using the *domain properties* section at the bottom of Figure 7. Domain properties are stored in a database of all domain properties associated with one 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 an existing property from the database or adds a new one to it. Figure 8 shows the management of the VANTAGE domain properties in REDEPEND – the list includes domain properties *DP10*, *DP11* and *DP12* featured in the example above. Finally, the satisfaction argument sheet can be exported to Microsoft Word to provide a more widely used documentation format for sharing within the project.

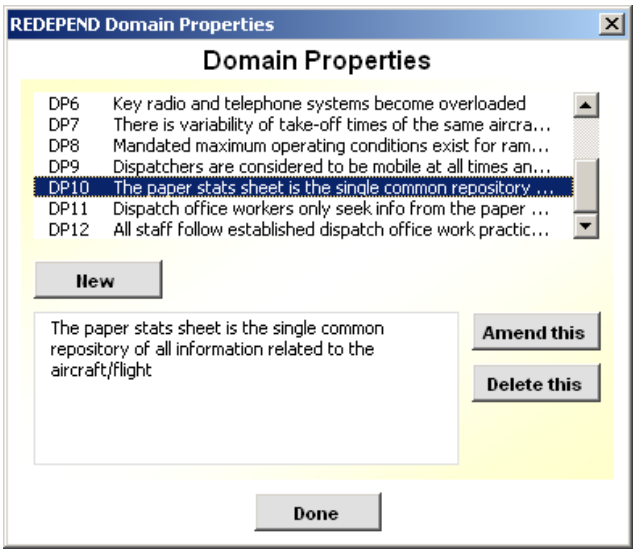


Figure 8: Domain property management in REDEPEND for all of the satisfaction arguments contained in the VANTAGE system model

4.2 Completing the Functional Requirement-SR Model Matrix

An analyst completes the functional requirement-SR matrix using a spreadsheet embedded in REDEPEND. Part of the matrix developed for VANTAGE is shown in Figure 9. An analyst copies functional requirements into the left column, then REDEPEND automatically generates the other columns with tasks and resources from the actors in the SR model. The analyst then completes the matrix by inserting "+" or "–" values to indicate compliance or non-compliance between the functional elements and the requirements. 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 9 demonstrates how an analyst can toggle quickly between a selected matrix row and the corresponding SR model element (*maintain paper stats sheet* task) on the diagram. We consider such model navigation is essential to support the analysis of large systems, such as those reported in our case studies, and to improve the usability of our method.

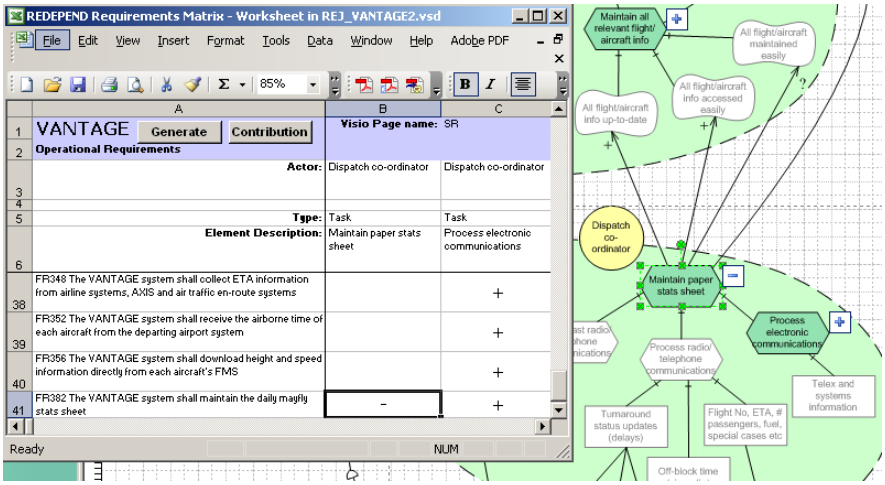


Figure 9: An example of mapping VANTAGE software requirements from the matrix to *i model elements using REDEPEND**

4.3 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 select a requirement in the matrix and run an automatic procedure to tag the impacts on affected tasks and resources in the SR model. Impacts are depicted on a model as a compliant "+" or non-compliant "–", whilst non-impacted elements are greyed out. Having selected an element with a detraction, REDEPEND can then be used automatically to detect and tag other elements that are *potentially non-compliant* by using propagation heuristics to follow the *i** links upwards, as reported in Section 3.4. Propagation impacts across task decomposition and dependency links are automatically determined in REDEPEND with *non-compliant* elements tagged with an "*X*", but propagation across means-end links are tagged as *undecided* "?" as these need analyst input.

Figure 10 shows the results of one walkthrough using the *maintain paper stats sheet* task. This task was detracted by the requirement *the VANTAGE system shall maintain the daily mayfly stats sheet*. The first level of propagation only included means-end links, therefore all propagated elements were tagged as *undecided*. Analyst judgment was then needed to decide whether the elements were *non-compliant* based upon domain properties and explanations using the satisfaction argument sheets. In this example, the *all flight/aircraft info accessed easily* soft goal was tagged *non-compliant* in light of the associated satisfaction argument, i.e. given the introduction of the VANTAGE system, the paper stats sheet may no longer be the single repository of all information related to the aircraft/flight; dispatch office workers may seek aircraft/flight info from sources other than the paper stats sheet; and established dispatch office working practices related to the upkeep of the stats sheet may no longer be suitable.

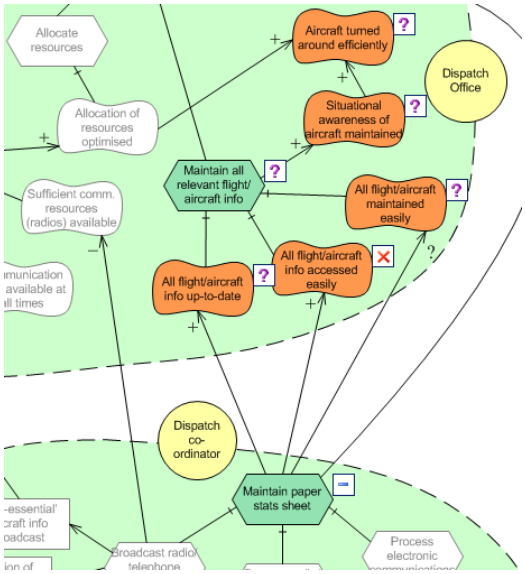


Figure 10: An example of exploring propagation impacts from a *non-compliant* task in the VANTAGE model using REDEPEND

4.4 Evaluation of the Method and Tool

Having developed the PiLGRIM method and implemented a new version of REDEPEND to support it, we sought to evaluate both. To do this we adopted a case study approach to explore the five claims summarized in Section 1.1. The claims were evaluated through two case studies, one in the area of air traffic management and the other in patient surveillance. Table 1 reports the claims and evidence we sought to collect to evaluate each of the claims.

Table 1: The five claims for the tool and procedures, and the evidence we sought to collect to evaluate these through the two case studies

Claims for the tool and procedures		Evidence to collect
1	The effective conceptual integration of *i** SR models and satisfaction arguments	Analysts use domain knowledge effectively to construct the *i** models; The inclusion of relevant domain knowledge in the specification of each new system
2	A requirements analyst can use the procedure with which to analyse the impacts of software system requirements on system-wide goals effectively	A requirements analyst is able detect valid impacts on goals and soft goals from requirements statements
3	A requirements analyst can use the software tool support with which to exploit the new procedure effectively	The level of coherent tool support for the procedures; The volume, frequency and severity of usability problems experienced by requirements analysts when using REDEPEND
4	A requirements analyst can use the procedures and tool to model large socio-technical systems with *i** effectively	The models and specifications of complex and large socio-technical systems produced by the analysts using PiLGRIM and REDEPEND
5	A requirements analyst finds the PiLGRIM method is useful and usable	The volume, frequency and severity of usability problems experienced by analysts when using PiLGRIM in the REDEPEND tool; The claimed benefits of PiLGRIM and REDEPEND against the effort required to use the procedures

		and tool made by requirements analysts

Of particular interest to us were the fourth and fifth claims that the PiLGRIM method scales to large and complex socio-technical systems and is usable by requirements analysts. Proving that the PiLGRIM method scales and is usable would enable us to follow the next stage of our research – to investigate if our method provides wider benefits to an organisation.

Our first case study in air traffic management sought to provide a proof of concept of the PiLGRIM method, with our second case study in patient surveillance taking this work forward with evaluation of a set of preliminary results. We report these studies in the following two sections.

5. Case Study One: NATS

For our first case study, proof of concept of the process and tool extension was investigated using an airspace infringement detection solution. The study took place at NATS, the UK's national air traffic service, and applied our tool-based extension to existing NATS requirements processes to support safety-related requirements analysis and specification (reported in an earlier form in [24]). Safety-related requirements processes were extended with *i** modelling supported with satisfaction arguments, embedded in an extended version of REDEPEND.

5.1 The Domain

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

Infringements of controlled airspace by unknown aircraft presented a significant risk to NATS. The safety need was to detect and bring to the controllers' attention infringements by unknown aircraft into controlled airspace. The system in place at the time relied on controllers noticing unknown aircraft entering controlled airspace when monitoring the radar display. For aircraft that transponded a Secondary Surveillance Radar (SSR) code, the Short Term Conflict Alert (STCA) system provided an alert about potential loss of separation (the safe distance between two aircraft operating in the same area) 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.

The Controlled Airspace Infringement Tool (CAIT) was to be a new safety net tool to provide controllers with more timely warnings of controlled airspace infringements by aircraft. The intention was that CAIT would 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.

5.2 The *i** models

For the first stage of the PiLGRIM method the authors worked with NATS domain experts to produce *i** models of the airspace infringement detection solution. One SD model and one SR model of key actors, goals, soft goals, tasks and resources were produced using the extended versions of *i** and REDEPEND. The *i** models were developed and validated in 6 half-day meetings over a 4-month period. During each meeting one 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 one-day observation of work at the London Terminal Control Centre. The SR model reflected the scale of the airspace infringement problem with 25 actors represented, 15 of which were expanded to include the actors' internal process elements. The components of the SR model are represented numerically in the Table 2. As one of the largest *i** models the authors have developed, its size means we cannot provide a single readable version of the model in this paper (a readable version is available at [28]), so the next paragraph highlights important elements of the SR model shown in Figure 11.

Table 2: Numerical summary of components in the CAIT *i SR model**

SR actors	**25**	*Of which*	Expanded actors	15
SR model elements	**197**	*Of which*	Tasks	77
			Goals	22
			Soft goals	37
			Resources	61
SR model links	**299**	*Of which*	Dependencies	59
			Contributes-to soft goal	48
			Means-end	73
			Task decomposition	119

The actor representing the new CAIT software system, a small safety net tool on which requirements were specified in text form, is shown (A) in the model in Figure 11. CAIT is a relatively small part of infringement detection and has dependencies with other actors in the NATS integrated air traffic control system. Actors include *aircraft* (H) 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 *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*. 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 [14]. 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.

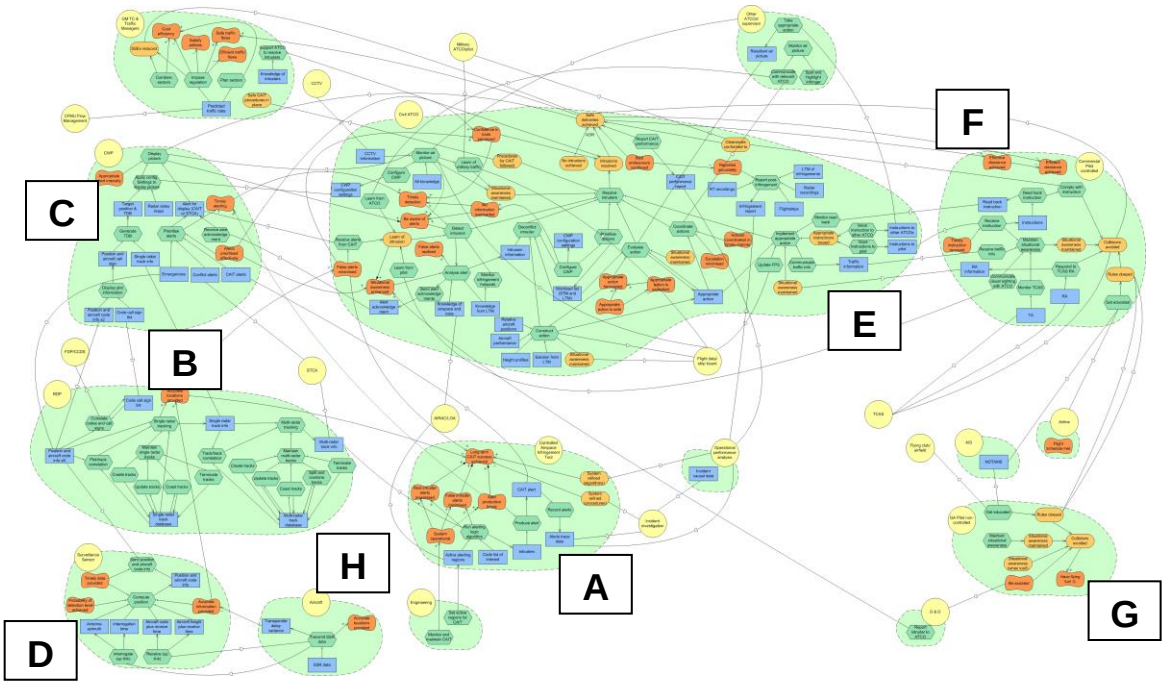
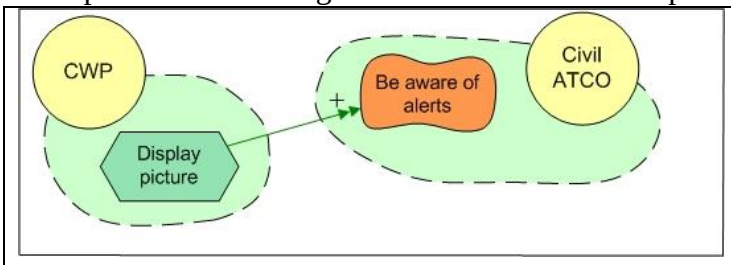


Figure 11: *i SR model of the CAIT system, developed using REDEPEND**

5.3 The Satisfaction Arguments

Having signed off the SR model as complete, the NATS analysts followed the second stage of the PiLGRIM method to develop integrated *i**-satisfaction arguments. The analysts elaborated 8 selected soft goal end-elements and their associated means-end links from the model based on their domain expertise. The authors generated the automatic elements of the satisfaction arguments using REDEPEND as described in Section 4.1. Basic training and instruction was then provided to the experts to manually complete the satisfaction arguments with domain properties. Each resulting argument contained, on average, 2 refinement means, 1 specification means and 3 domain properties. The database associated with the SR model contained a total of 22 domain properties, with no reuse of CAIT domain properties during the development of the satisfaction arguments. The lack of reuse was most likely due to the low number of satisfaction arguments developed during the NATS study – only eight due the limited availability of domain experts.

Figure 12 shows one of the selected means-end links and its associated satisfaction argument. In this instance the end-element represented current human work required to change due to requirements on CAIT, and the means-element represented a current system also required to change due to requirements on the introduced system. The double-headed arrow specified a MAKE contributes-to soft goal link – the contribution of the completed *CWP display picture* task was positive and sufficient to satisfice the *civil ATCO* soft goal *be aware of alerts*. The argument was important for the introduction of *CAIT* even though *CAIT* did not form any element of the satisfaction argument. The domain properties *the ATCO is monitoring the air picture* and *alerts are only communicated through the air picture* rely on specifications and work practices beyond the scope of the CAIT specification, and are therefore assumptions rather than goals that CAIT would be specified to achieve.



Satisfaction argument

Actor: Civil ATCO
End-element [type]: Be aware of alerts [Soft goal]
Specification means-element [type] [actor] [link]: Display picture [Task] [CWP] [Make]
Domain properties:

- The ATCO is monitoring the air picture
- Alerts are only communicated through the air picture

Explanation: 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.

Figure 12: An example of a means-end link and its associated satisfaction argument taken from the CAIT system model

5.4 Developing the Requirements-*i** Model Element Matrix

The analysts were given a specification document complete with 31 requirements on the CAIT software that we inserted into the left-hand column of the matrix to begin the third stage of our method. The top row of the matrix was automatically populated in REDEPEND with 77 tasks and 61 resources from the SR model. The 31

requirements were developed independently by a CAIT software team, and were not accessed during the *i** modelling to reduce the possibility of bias in assessing requirement impacts.

Two NATS requirements analysts applied the PiLGRIM method as described in Section 3.3, mapping the requirements to the tasks and resources using compliant/non-compliant values. Figure 13 reports the compliant and non-compliant values for two of the requirements. The requirement *CAIT alerts shall be logged automatically by the system* will, if satisfied, be compliant with the successful completion of the task *record alerts*, but will be non-compliant with the availability of the *civil ATCO* resource *long-term memory of infringements* – as ATCOs would no longer be involved in the conscious process of filing manual infringement reports. The requirement *it shall be possible to set active regions for the CAS infringement alerts* will, if satisfied, be compliant with the successful completion of the *engineering* task *set active regions for CAIT*.

Functional requirement	CAIT: Record alerts [T]	Civil ATCO: L-T Memory of infringements [R]	Engineering: Set active regions for CAIT [T]
CAIT alerts shall be logged automatically by the system	+	–	
It shall be possible to set active regions for the CAS infringement alerts			+

Figure 13: A part of a functional requirement – SR model matrix for the CAIT system. NB: requirements are not original CAIT requirements for confidentiality reasons

Alas, due to finite resources and time the NATS case study ended before the matrix was completed, therefore the analysts were unable to follow the fourth stage of the PiLGRIM method to determine the impact of *non-compliant* tasks and resources on the goals and soft goals in the model. Although the final stage of our method was not attempted by the analysts in this case study, we were able to use the data from the first three stages for assessing claims 1, 3 and 4.

5.5 Other Observations

The NATS analysts were uncertain how to deal with means-end links contributing negatively to soft goals, as the PiLGRIM method and its rationale did not describe what to do. As a result, the analysts recorded the assumptions behind the negative contributions captured in the air traffic domain as a separate exercise. This led the analysts to consider assumptions concerned with the impact of new operational procedures, and effectiveness versus efficiency – assumptions of significance for the introduction of the new software system.

Whilst no major usability problems with REDEPEND were reported during the *i** modelling, an issue arose concerning the large tool-based requirements matrix and the ease of completion. The NATS analysts felt it would be useful to have a built-in tool feature to print out the SR diagram and corresponding sections of the requirements matrix, to enable them to generate by hand a first-cut matrix prior to entering data into the tool. We addressed this issue and the effort involved in completing a REDEPEND requirements matrix later on for the second case study in Section 6.4.

5.6 Conclusions

Despite the large scale of the infringement detection problem the NATS team used PiLGRIM and REDEPEND to develop *i** models and satisfaction arguments successfully. The arguments described domain properties missing from the original SR model but deemed important by NATS analysts. However the impact analysis was not completed due to time constraints, with implications for evaluating claims 2 and 5. Table 4 provides a summary of the evidence collected from this case study to evaluate the remaining three claims. Results provided some empirical evidence for our conceptual integration of *i** modelling and satisfaction arguments, the usability of our tool, and the scalability of PiLGRIM.

Table 4: Three of our claims and a summary of the evidence we collected to evaluate these through our first case study

Claim		Evidence
1	The effective conceptual integration of *i** SR models and satisfaction arguments	• The NATS analysts were successful in developing and reasoning satisfaction arguments for *i** means-end links; • Domain properties that were not explicitly represented in the *i** semantics were added. NATS analysts considered these domain properties important to the model's completeness for the specification of the CAIT system
3	A requirements analyst can use the software tool support with which to exploit the new procedure effectively	• REDEPEND supported a consistent application of the first 3 stages of the PiLGRIM method successfully. The final stage of the method was not evaluated in this study; • No usability problems were reported during the *i** modelling, but the analysts recommended improvements to the usability of the requirements matrix
4	A requirements analyst can use the procedures and tool to model large socio-technical systems with *i** effectively	• The NATS project team were able to develop large *i** models of the CAIT system successfully using REDEPEND

So, having demonstrated the first 3 stages of our method and explored the potential of determining goal and soft goal impacts, we decided to undertake a second case study to obtain a more complete set of research results for our approach, and in particular undertake impact analyses.

6. Case Study Two: Health Protection Agency

We used our second case study to investigate the impact of requirements changes on system-wide goals using *i** models and satisfaction arguments. The study took place at the Health Protection Agency (HPA) in the UK, where our approach was applied to analyse the impact of change request requirements raised for the HIV/AIDS Patients monitoring system (HAP). The same REDEPEND tool and procedures from the NATS case study were applied. The analysis in this case study was undertaken by an employee of the HPA who is also an author of this paper (full details of the study are available in [5]).

6.1 The Domain

The HAP system monitors the impact of HIV on public health in the UK by running a system originally designed in September 1982 following early reported cases of HIV infection [4]. This system has evolved and improved over the years, and now is widely regarded as one of the most comprehensive Public Health Surveillance (PHS) systems in the world, covering key features recommended by the Guidelines Working Group [6]. Evolution of HAP has been informed by feedback from HAP users, technological advances, updates in public health legislation and evolution of the HIV epidemic. Updates to the system are managed through change requests raised by users and implemented by the Centre for Infections Software Development Unit (SDU).

HAPv3, the latest version of HAP, was commissioned and built in 2003 to combine the functionalities of several peripheral applications previously developed to meet evolving requirements of HAPv2. These peripheral applications resulted in duplication of requirements (and therefore functionality), confusion amongst users (for example due to the number of different screens for the same process), as well as increasing the chances of conflicting requirements between applications.

In 2008, as with previous versions, HAPv3 was once again perceived to be complicated by the development of further peripheral applications. Although the change request procedure had been effective in solving individual problems, there was no method in place to monitor the impact of each change on the wider system goals. Lack of this middle ground resulted in users having difficulties explaining their underlying needs, and engineers having difficulty describing the impact of technical solutions. Hence we applied PiLGRIM to the problem, undertaking requirements change analysis on important goals and soft goals.

6.2 The *i** Models

Modelling for the first stage of our method was performed by one analyst with specialist knowledge of HAP using information obtained from departmental documentation and five years experience with the system. One SD model and one SR model were developed using REDEPEND version 5, the same modelling tool used during the NATS project. As with the previous project, the HAP requirements were developed independently and were not used as inputs into the *i** models. The models were discussed through a series of meetings with HAP users to confirm the context of dependencies being modelled, and with the analysts who worked with NATS to validate the use of *i** syntax. The process took approximately 3 person days to complete. The SR model described 12 actors, 5 of which were expanded to include 102 internal process elements and 130 links, as detailed in Table 5. The next paragraph highlights elements of the model shown in Figure 14.

Table 5: Numerical summary of components in the HAP *i SR model**

SR actors	**12**	*Of which*	Expanded actors	5
SR model elements	**102**	*Of which*	Tasks	55
			Goals	18
			Soft goals	19
			Resources	10
SR model links	**130**	*Of which*	Dependencies	19
			Contributes-to soft goal	32
			Means-end	18
			Task decomposition	61

The actor representing the *HAP* software system, on which change request requirements were specified in text form, is shown (A) in the model in Figure 14. The software application is at the core of all UK HIV/AIDS patients monitoring and reporting, and validates links between databases and stores information on new HIV diagnosis records. It undertakes tasks such as *merge linked records* and *validate data* to achieve soft goals such as *patient record completeness maximised*. HAP depends on resources produced by other actors and managed by the *information officer* (B), for example *death reports* from the *ONS*. *Inputters* (C) enter paper records into databases, and depend on *HAP* to *display input errors* and *display notifications with similar patient IDs*. *Scientists* (D) perform epidemiological analysis and interpret data to keep the public informed and up-to-date, and *research nurses* (E) meet with patients and clinicians to identify and classify probable infection routes and other patient information that is unclear on the reported form.

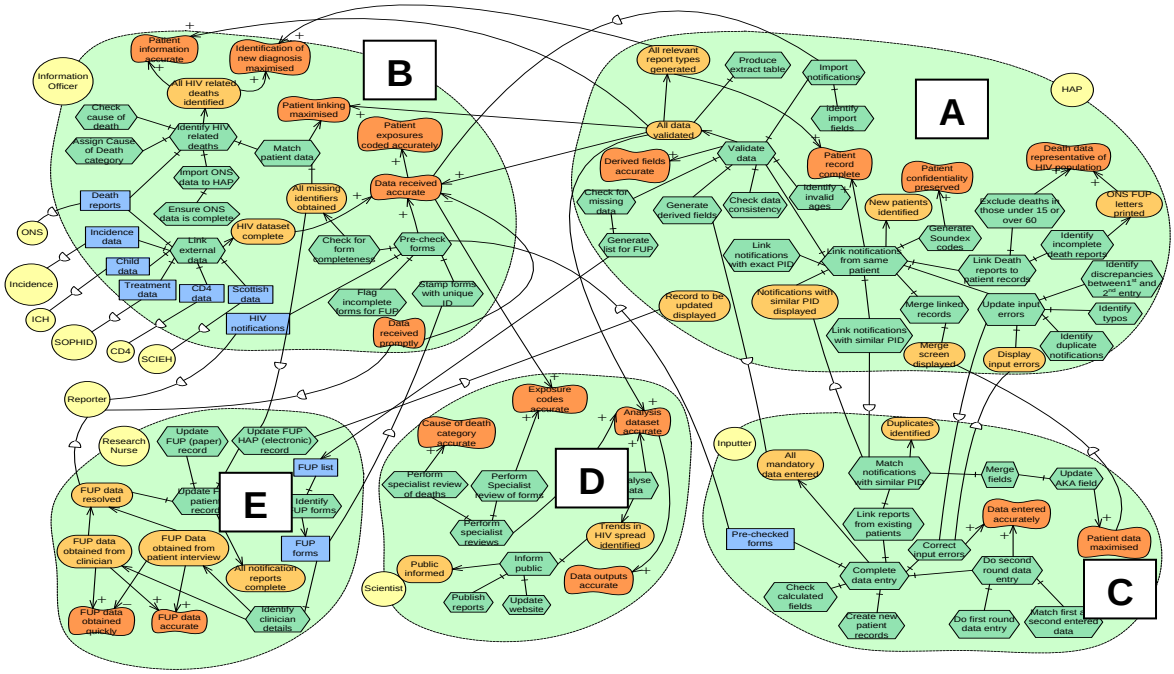


Figure 14: *i SR model of the HAP system, developed using REDEPEND**

6.3 The Satisfaction Arguments

The HAP analyst specified satisfaction arguments for soft goals and goals in line with stage two of the PiLGRIM method. He also extended the method to apply satisfaction arguments to tasks and resources. However, the intention of PiLGRIM

was to assess the impacts on end goals by using the means specifications and associated domain properties as part of the complete end argument. So, for example, the satisfaction argument for the means task *identify HIV related deaths* should have been allocated to its end goal *all HIV related deaths identified*. Therefore, we include these results only as evidence of the analyst's ability to develop and reason satisfaction arguments for elements and associations in the *i** model.

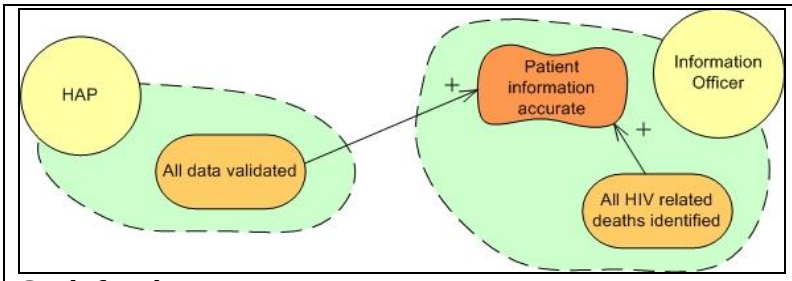
Unlike in the NATS study, domain property specification was performed using the REDEPEND tool throughout the modelling and during impact analysis of change requests. In total, 22 satisfaction arguments were developed for the SR model – 13 for soft goals, 3 for goals, 5 for tasks and 1 for a resource. In contrast to the CAIT satisfaction arguments domain properties were reused. Of the 40 unique domain properties specified 15 were reused, leading to a total of 60 uses (1 domain property was used four times, 3 used three times and 11 used twice). The HPA analyst reported reuse of domain properties in REDEPEND to be both useful and straightforward. Table 6 summarises the make-up of the 22 satisfaction arguments.

Table 6: Cumulative total and average number of refinements, specifications and domain properties in the 22 HAP satisfaction arguments

	Refinements	Specifications	Domain properties
Total	34	10	60
Average (mean)	2	0	3

Domain property specification was incremental, with 27 specified during the production of the SR model and another 13 added during the matrix completion and impact analysis. The analyst reported that applying the final two stages of the PiLGRIM method revealed additional domain properties that were overlooked during the initial modelling phase.

An example satisfaction argument developed for the *information officer* soft goal *patient information accurate* is shown in Figure 15. The end-element and refinement means-element represented goals related to current human work, where the achievement of the *information officer* goal *all HIV related deaths identified* contributed positively to the *patient information accurate* soft goal. The specification means-element represented a goal, *all data validated*, that was impacted on by change request requirements on the HAP system. The argument stated that validating records and checking for completeness enhanced information accuracy, given that certain domain properties held true, e.g. *patients give accurate information to clinicians*, an assumption as a database redesign would not able to achieve this as a goal.



Satisfaction argument

Actor: Information Officer
End-element [type]: Patient information accurate [Soft goal]
Refinement means-element [type] [link]: All HIV related deaths identified [Goal] [Some+]

Specification means-element [type] [actor] [link]: All data validated [Goal] [HAP] [Some+]
Domain properties:

- Patient data reported from reporters are accurate
- Patients give accurate information, such as dates of birth, to clinicians
- The same Soundex code algorithm is used to Soundex surnames between different reporting sites to ensure that they generate the same codes for identical patients which can then be linked
- Information present in the death record can be used to validate record

Explanation:
Patient records are collected from different data sources (laboratory records, clinical records and death reports), which are linked for individual patients to provide a complete patient record and to validate record elements. Each record needs to be checked for accuracy at different stages and processed in the same way to enhance the ability to link between datasets and improve information accuracy.

Figure 15: An example satisfaction argument taken from the HAP system model

6.4 Developing the Requirements Matrix

During the third stage of the PiLGRIM method the HPA analyst took 29 requirements and the current set of unimplemented change requests raised by HAP users, and inserted them into the requirements matrix for impact analysis. The top row of the matrix was automatically populated in REDEPEND with the 55 tasks and 10 resources from the SR model. The analyst then added compliant and non-compliant values in the matrix for each change request requirement, as described in Section 3.3. The analyst identified a total of 100 compliant values and 9 non-compliant values (see Table 7), demonstrating that most change requests had positive impacts on the HAP system. Each change request impacted directly on average on three tasks and/or resources in the SR model. Four requirements did not impact on any tasks or resources.

The analyst needed 2-5 minutes per impact assignment to a task or resource. Due to its scale, it took close to one person day to complete the matrix, in spite of the analyst's experience with the system and knowledge of the *i** models. Although no general usability problems with REDEPEND were reported, the analyst experienced that changing the model and/or requirements displaced previously entered compliance values in the matrix. As such this limitation added to the effort required for matrix completion.

Table 7: Summary of compliant/non-compliant values in the HAP requirements matrix

Task/resource impacts	
Total number of compliant values	100
Total number of non-compliant values	9
Median number of task/resource impacts per requirement [interquartile range]	3 [4]

6.5 Determining Goal and Soft Goal Impacts

Once the matrix was completed, the analyst used REDEPEND to perform the goal and soft goal impact analysis as described in Section 3.4. The analyst also ran propagations on compliant elements to identify and verify the resultant compliant goals and soft goals. This complete analysis generated the 225 compliant and 29 non-compliant values reported in Table 8. Again analysis was not quick because the 29

non-compliant goals and soft goals required human analysis of satisfaction arguments taking on average 10 minutes per argument. In total one person day was used to complete the goal and soft goal impact analysis.

Table 8: Summary of impacts on goals/soft goals in the HAP model. NB: these are cumulative totals from the propagations applied for each of the 29 HAP requirements

Goal/soft goal impacts	
Cumulative total of goal/soft goal compliant values	225
Cumulative total of goal/soft goal non-compliant values	29
Median number of goal/soft goal impacts per requirement [interquartile range]	6 [11]

The propagation analysis focused on four requirements, listed in Table 9, that were non-compliant with tasks and resources in the matrix and therefore needed analyst input. The table shows the number of compliant and non-compliant tasks/resources, the number of compliant and non-compliant goals/ soft goals, and a description of the potential impact obtained from applying the methods.

Table 9: Selected results from the impact analysis of change request requirements on HAP

Requirement	**Task and resource compliance**	**Goal and soft goal compliance**	**Description/implication of impacts**
RE007 - Information officer shall be able to modify exposure codes	4 Compliant 1 Non-compliant	14 Compliant 4 Non-compliant	Has positive impact on Tasks of Information officer, Inputter, and Scientist **IF** modified codes are consistent. If not, this would lead to negative impact on these actor tasks. Maybe scientist who has specialist knowledge should do this. This ambiguity could be solved with a better formulated requirement such as **"***exposure codes shall be representative of all HIV transmission routes***"**
RE014 - HAP shall pass incomplete information from patients with only a lab report to the FUP database two months after lab report date	0 Compliant 6 Non-compliant	0 Compliant 6 Non-compliant	This negatively impacts the ability to identify FUP forms (cannot confirm FUP status until after 2 months have passed). This negative impact propagates across dependant goals and soft goals. Struggled a bit with assignment of detraction value maybe because requirement is non functional. It could be seen to enhance FUP process by alleviating the need to perform FUP on records that may be completed within two months.
RE016 - HAP shall be able to accept a large number of records	2 Compliant 1 Non-compliant	15 Compliant 19 Non-compliant	Small field sizes made data input difficult, so satisfying this requirement will enhance data input, but many records make it difficult for HAP to check for missing data. This shows how SR model can bring perspective to the new requirement in relation to other system tasks and goals (enhancing some, but detracting others)
RE029 - HAP shall be able to create death reports from multiple sources	3 Compliant 1 Non-compliant	3 Compliant 0 Non-compliant	Enhances ability to identify HIV related deaths, but it would be hard to check the actual cause of death if reports are in different formats

For requirement *RE029 - HAP shall be able to create death reports from multiple sources*, the analyst identified 1 non-compliant task, as shown in Figure 16 (the *check*

cause of death task in Stages 1 and 2). The analyst then ran the goal impact propagation which in turn led to 1 more *non-compliant* task, 1 *undecided* goal and 2 *undecided* soft goals, as shown in Figure 16 (stage 3). The impacts propagated across the means-end links were automatically tagged as *undecided* "?" as per the method. Therefore, the analyst assessed the satisfaction arguments associated with these *undecided* end-elements and rewrote them to achieve compliance, hence the zero value for goal/soft goal non-compliance in Table 9.

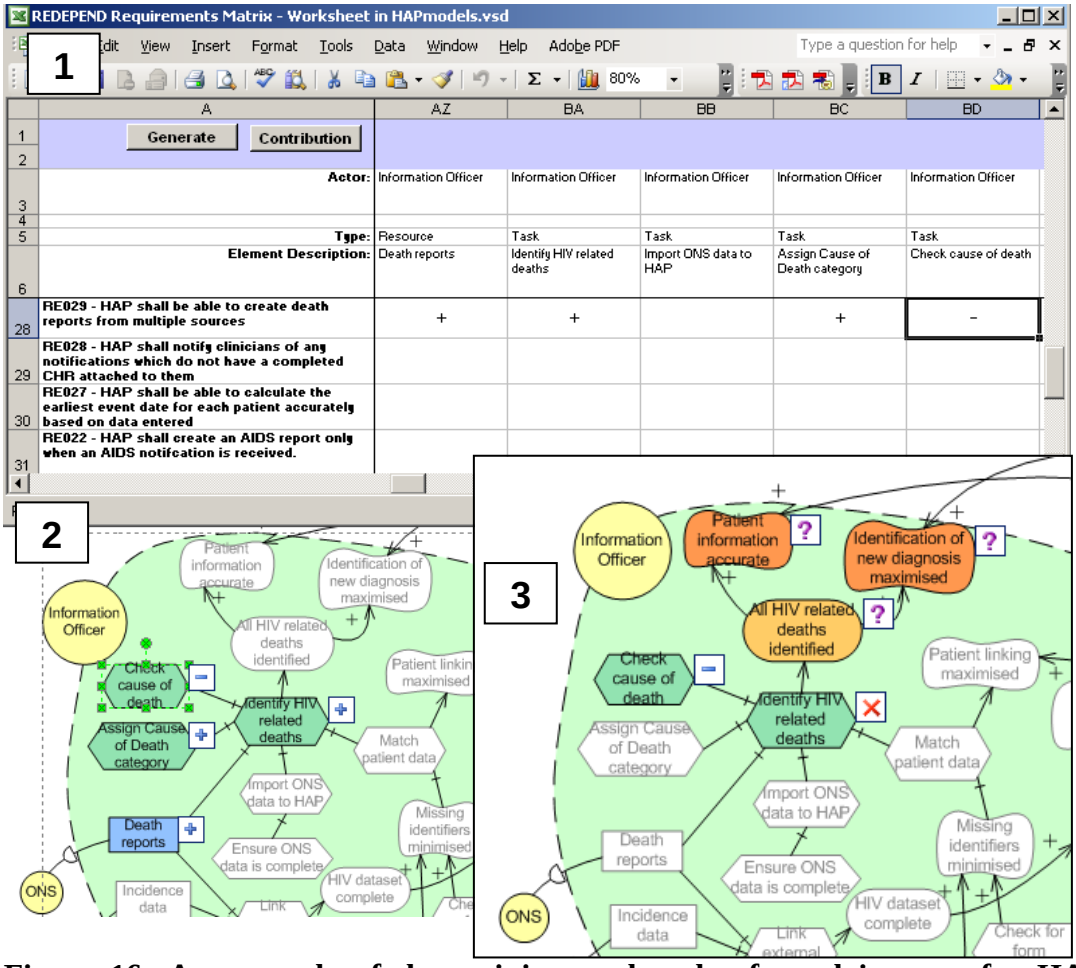


Figure 16: An example of determining goal and soft goal impacts for HAP, showing the requirements matrix and automatic propagation in REDEPEND

The application of the propagation heuristics did not invalidate the domain properties for most satisfaction arguments. One possible explanation for this was that the authors of the change requests had respected assumptions about the well-established HAP domain. Where potentially *non-compliant* goals and soft goals were identified through heuristics 5 and 6, it was possible to make them compliant by adding new domain assumptions to the arguments.

6.6 Unsupported Activities

The analyst also identified two activities that were not supported by the PiLGRIM method. The first was that the analyst was not able to handle mutually-dependent change request requirements as the method addresses requirements individually. For example, the change request requirement to notify a nurse of a deleted record has no impact unless the requirement to delete the record is also considered, i.e. the

notification simply would not be triggered without the dependent requirement also being implemented. This raised a future need for PiLGRIM to support the analysis of more than one requirement at a time.

The second was that the analyst was uncertain how to handle requirements that would alleviate the need for tasks or resources in the SR model. For example, the requirement that the notification for follow-up (FUP) would only be sent for new patients if the record was incomplete alleviates the need to do FUP on all new notifications. As such, this means that nurses would not need to do data follow up on new records *per se*, just records that were incomplete. Again, the analyst found matrix completion in such a case to be difficult.

6.7 Conclusions

The HPA analyst was able to develop one SD model, one SR model and satisfaction arguments linked to means-end links in the SR model using the extended versions of *i** and REDEPEND. He was able to discover domain properties in the satisfaction arguments incrementally during modelling and analysis tasks, which led to domain properties more relevant to goals and soft goals. Unlike in the first case study, the analyst also mapped 29 change request requirements to *i** model tasks and resources, then undertook impact analyses to show that 23 impacted positively on HAP system goals, 3 impacted negatively on these goals, and 3 revealed redundant tasks and resources. As a consequence these latter 6 requirements were reviewed and changed prior to their implementation.

Table 10 provides a summary of the evidence collected to evaluate the five research claims. We believe that results provided some empirical evidence for our conceptual integration of *i** modelling and satisfaction arguments, the usability of REDEPEND, and the scalability of the PiLGRIM method, and ultimately the usefulness of the PiLGRIM method. One caveat is related to analyst experience. A less familiar and/or experienced analyst might have taken more time to complete the requirements matrix.

Table 10: Four of our claims and a summary of the evidence we collected to evaluate these through our second case study

Claim		Evidence
1	The effective conceptual integration of *i** SR models and satisfaction arguments	▪ The HPA analyst was successful in developing and reasoning important satisfaction arguments; ▪ Domain properties that were not explicitly represented in the *i** semantics were added during and after development of the SR model. The HPA analyst considered these domain properties important to the model's completeness for the specification of the HAP system
2	A requirements analyst can use the procedure with which to analyse the impacts of software system requirements on system-wide goals effectively	▪ The HPA analyst was able to detect positive and negative impacts on requirements on system-wide goals and soft goals; ▪ However analysts encountered difficulties when assessing impacts of requirements that were mutually-dependent and led to redundancies in the *i** SR model

3	A requirements analyst can use the software tool support with which to exploit the new procedure effectively	▪ REDEPEND supported a consistent application of all 4 stages of the PiLGRIM method successfully; ▪ No usability problems were reported during the *i** modelling, but the analyst recommended improvements to the usability of the requirements matrix
4	A requirements analyst can use the procedures and tool to model large socio-technical systems with *i** effectively	▪ The HPA analyst was able to develop large *i** models of the HAP system successfully using REDEPEND
5	A requirements analyst finds the PiLGRIM method is useful and usable	▪ The HPA analyst was able to show the impacts of all the change request requirements on the HAP system-wide goals; ▪ The effort involved was approximately 5 person days. It was too early to assess the wider benefits to the organisation

7. Related Work

The PiLGRIM method represents a pragmatic integration of existing requirements concepts and techniques, with two sources of ideas being of particular influence in the development of the method – satisfaction arguments, e.g. [9], and qualitative goal analysis procedures such as [27]. Our motivation for PiLGRIM was to develop a useful and useable tool-supported method that would successfully engage analysts with these concepts and techniques. Therefore, in this section, we will look at related work in the context of this aim.

7.1 Satisfaction Arguments in Requirements Methods

The original conception of a satisfaction argument from Jackson [17, 40] has already been incorporated into requirements methods such as problem frames [e.g., 18, 7, 8], rich traceability [e.g., 3, 9] and KAOS [34, 35]. We look at each area in turn.

Jackson's problem frames approach [18] focuses on the software (machine) and the application domain, and includes a problem context that represents the machine to be built, the various problem domains in the application domain, and the interfaces between them. A problem diagram, which uses the problem context, introduces the requirements to bring about certain effects in the problem domains. However, this approach to modelling a single system (machine) does not translate well to socio-technical systems that have multiple systems and actors. The approach also includes concepts for describing recognisable classes of problems. Similar to a design pattern, these classes are abstractions of repeating problems. Again, these do not transfer and scale well to a complex socio-technical system due to their simplicity.

Haley et al [7] use satisfaction arguments to show that a future realised system can satisfy its security requirements. Their approach is based on problem analysis and uses an approximation of Jackson's problem frames diagrams to discover security requirements. Their approximation does not attempt to identify a particular problem class, but enters phenomena and requirements into a system problem diagram that expresses the interaction of domains from a security perspective. This approach does not attempt to analyse the wider development problem and phenomena across a wide range of domains. Formal arguments, based on claims about domain properties, are coupled with informal arguments that justify the claims [7]. The arguments are captured in a text form as this is the most expressive and natural way of representing them. However, Haley et al [8] report that the text representation of an argument can become difficult when the argument is complex. For example, project members in their case study were more comfortable using a less expressive graphical form of the informal arguments. Therefore, tool support for translating between textual and graphical forms of the arguments would have been helpful. We sought to take these concerns of complexity management and visualisation into account for the PiLGRIM method.

In [3], Dick uses a simple textual representation of a satisfaction relationship for a customer requirement linked to artefacts and associated rationale. He recommends 'appropriate' tool-based support to construct a tree of related artefacts, and a configuration management tool to define changes through the design layers. REVEAL, described in section 2.3, supports different representations of satisfaction arguments for simple use in requirements projects. For example, Mavin [26] demonstrates a straightforward tabular structure to link specifications and domain properties to satisfaction argument text when applied to requirements on rail rolling stock. Also, Hammond et al [9] present a graphical representation of rich traceability

with similarities to the AND/OR graphs used in the KAOS method. Their approach has a more informal representation than KAOS and also shows an explicit justification of how and why a statement is achieved. However, despite this similarity, REVEAL still lacks the use of explicit goal modelling techniques.

KAOS [2] is a well-established goal-based requirements engineering approach that has been applied in multiple industrial projects over a period of nearly twenty years [35]. van Lamsweerde states in [35] that an important, often neglected part of a requirement engineer's role is to specify satisfaction arguments. As mentioned earlier, KAOS uses AND/OR goal refinement graphs to represent such rich traceability. The refinement is elaborated with domain properties in order to determine whether its sub-goals are sufficient to establish satisfaction of the parent goal [20]. However, AND/OR trees expressing hierarchy do not always accurately reflect the complex nature of contributions between goals, especially when considering heterogeneous actors in socio-technical systems.

Another well-established goal-based approach is *i**, our method of choice, as introduced earlier in section 2.1. *i** is a more lightweight method than KAOS and contains a vivid visual representation of actor boundaries. Establishing boundaries of socio-technical systems and placing goals into an organisational context is a key requirements task. A major strength of *i** is its ability to represent a clear overview of these important concepts. Also, goal expression is helped in *i** through the modelling of process elements, such as the ATCO task model developed in our first case study.

7.2 Qualitative Goal Analysis

The PiLGRIM impact analysis procedure shares some characteristics with the NFR Framework [27]. As stated in [27], quantitatively measuring a new or incomplete software system is an even more difficult task than that of measuring the final product. The aim is to rationalise the development process and justify design decisions early on in the project. This also applies to the PiLGRIM method, as the soft goals expressed at the early development stage are, by definition, not measurable. There is also a similarity with the NFR labelling procedure, which determines the status of a goal using four main labels: *satisificed* (the software is expected to satisfy the goal within acceptable limits), *denied* (unsatisficeable or "unsolvable" in problem solving terminology), *conflicting* (both deniable and satisficeable) and *undetermined* if neither. These labels are similar to *compliance* and *non-compliance* used in the PiLGRIM method. However, NFR labelling is a more simplistic approach where subjective judgments are made to determine the impact of a design decisions on the status of a goals. PiLGRIM uses satisfaction arguments, rather than judgement alone, to determine whether actual software requirements are compliant with the socio-technical system goals.

Horkoff et al's *i** evaluation procedure reported in [10] builds upon the original NFR Framework, described above, and shares further characteristics with the PiLGRIM method. Their qualitative reasoning approach applies propagation rules to *i** models to apply an extended set of NFR labels to represent the level of evidence towards the qualitative satisfaction and denial of model elements. Both the evaluation procedures use *i** semantics to propagate values of elements through an *i** model. However, our use of integrated *i**-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 formalise documentation of contextual knowledge in the *i** evaluation procedure [10].

7.3 Tool Support for the *i** Framework

A number of *i** tools have been developed, most of which are and listed and described in [15], for example the Organization Modelling Environment (OME) java application developed at the University of Toronto [30]. The OME tool provided the first modelling environment for the *i** framework, and included a graphical interface for developing models, usability features such as the facility to expand the SD model to show the internal elements of the SR model, and computer-aided analysis support [30]. However, the tool has not been shown to support large-scale *i** diagramming effectively, with the largest reported model containing approximately 100 elements [16]. In light of the limitations of the OME tool, REDEPEND was developed to support scalable *i** modelling and tool-supported analysis procedures as presented and demonstrated in this paper. For example, the scale of *i** modelling undertaken by the NATS analysts is reflected in Table 2, and the successful application of our procedures was demonstrated by the results of the HPA case study.

Subsequent *i** tool development of interest includes the Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool (OpenOME), an Eclipse-based open source evolution of the OME tool [13]. In particular, the tool has been specified to support Horkoff et al's *i** evaluation procedure [10] described above. The propagation procedures have been evaluated in a series of user studies reported in [11, 12], covering the propagations of qualitative labels and visualizations to improve usability. Other related OpenOME developments include the Open Requirements Engineering tool, which integrates Jackson's problem frames [18] with the *i** framework by hyper-linking model elements to arguments [39]. Whilst both of these recent OpenOME developments have similarities with our PiLGRIM approach, it remains that no tool apart from REDEPEND supports both propagation of change requirements and satisfaction arguments. We claim novelty for our tool-supported approach through this integration and its evaluation in two industrial case studies.

7.4 Extensiveness of the PiLGRIM Method

Finally, it is worth considering how general our approach is and whether our *i**-based extension could be applied to other goal modelling frameworks. We will use KAOS as an example, as it includes the main concepts modelled in the PiLGRIM method and there are obvious comparisons to be drawn.

KAOS uses four main models – the goal, responsibility, operation and object models. The method begins with identifying goals which are decomposed into subgoals modelled as directed graphs, which means a given goal can appear in more than one higher-level goal decomposition [32]. This notion would fit with PiLGRIM in that goals, especially non-functional goals, do not necessarily appear once in a simple goal-tree structure. However, we have found in our reported case studies that hierarchy does not always accurately reflect the complex nature of contributions between such qualitative goals. KAOS also includes non-functional goal decompositions that describe positive and negative contributions. These decompositions can be compared to the means-end link in *i** and PiLGRIM method extension. Similar to PiLGRIM, each goal can be refined with domain properties that are relevant to the application domain and used to prove the completeness of the refinement. KAOS includes an additional notation for expectations, for example behaviour expected of human actors, which we document under domain assumptions

for simplicity. KAOS goal refinement also includes requirements which are defined as goals that have been allocated under the responsibility of an actor. The associated actors are connected to the requirements and feature in their own responsibility model [32]. For PiLGRIM, the existing requirements specification, or set of change requirements, could be directly specified as requirements in KAOS.

The second KAOS model contains each actor's responsibility diagram that represents all of the requirements and expectations that they are responsible for [32]. This model has similarities with the actor boundary in *i**, but does not contain goals, tasks and resources. Instead KAOS uses two further models to represent objects and operations. The object model is compliant with UML class diagrams and includes entities, similar to *i** resources, which represent independent passive objects [32]. Similar to the task structures created in *i**, KAOS includes an operation model which describes all the behaviours that agents need to fulfil their requirements. Operations are applied to objects defined in the object model which can be created, changed or activated by other operations. In KAOS requirements can be operationalized by objects and operations [32]. For PiLGRIM, detailed analysis of agent behaviours is not required for the satisfaction arguments and propagation procedure, therefore reflecting operations in a task model is sufficient. It is possible, however, that further analysis beyond the method could benefit from the KAOS approach.

In conclusion, KAOS includes all of the necessary concepts and constructs, and more, to incorporate the PiLGRIM method but the main difference with *i** is the graphical representation. PiLGRIM requires just one SR model unlike KAOS which is likely to require a few models to represent all the concerns. We believe that the clear representation of actor boundaries containing all of the constructs and associations in *i** provides analysts with a more usable viewpoint from which to apply the PiLGRIM method. KAOS which includes additional, and useful, concepts such as events and obstacles lends itself better to automated reasoning. This is not the intention of the PiLGRIM method. Additional concepts, details and more complex visualizations present a trade off with usability when considering the time and effort required by analysts to learn and understand the method.

8. Research Contributions Revisited

We revisited the five claims for research contributions made in Section 4.4 to determine which were supported by the combined evidence obtained from both case studies. Results are summarized in Table 11 and provide evidence to support three claims fully and two partially. No claims were rejected. Each claim is discussed briefly in turn.

Table 11. Summaries of reported evidence to accept fully and partially the 5 claims made for the PiLGRIM method

Claims for the tool and procedures		Supported	Reported Evidence
1	The effective conceptual integration of *i** SR models and satisfaction arguments	Yes	Analysts in both case studies used domain knowledge effectively to construct the *i** models; Domain knowledge was included effectively in the specifications of the new systems in both case studies

2	A requirements analyst can use the procedure with which to analyse the impacts of software system requirements on system-wide goals effectively	Partially	The analyst in the HAP case study was able to detect positive and negative impacts of requirements on system-wide goals and soft goals; However resources were not available to complete the procedure in the NATS case study
3	A requirements analyst can use the software tool support with which to exploit the new procedure effectively	Yes	REDEPEND supported the application of at least 3 of the 4 stages of the PiLGRIM method successfully in both case studies; No usability problems were reported during the *i** modelling in both case studies
4	A requirements analyst can use the procedures and tool to model large socio-technical systems with *i** effectively	Yes	The analysts in both case studies were able to develop large *i** models successfully using REDEPEND
5	A requirements analyst finds the PiLGRIM method is useful and usable	Partially	The HPA analyst was able to show the impacts of all the change request requirements on the HAP system-wide goals; However the NATS analysts were unable to complete the impact analyses due to lack of resources

We argue that the PiLGRIM method delivered effective conceptual integration of *i** and satisfaction arguments (*Claim-1*). Analysts in both case studies were able to integrate domain knowledge into *i** models and requirements specification using the conceptual model reported in Figure 3. Results did not provide any rationale for changing the conceptual model. Therefore we argue that we resolved the first *i** weakness reported in section 2.2, to support and capture reasoning about *i** means-end links.

The PiLGRIM method also provided an effective procedure with which to analyze the impacts of 29 change request requirements on goals and soft goals in the second case study (*Claim-2*). However, the resources needed to undertake this analysis, and the failure to complete in the first, raises issues about the procedure's cost-effectiveness.

The PiLGRIM method provided effective software tool support with which to exploit the new procedure (*Claim-3*). Analysts in both case studies were able to implement the first three stages successfully, and the HAPS analyst had sufficient resources to undertake the fourth stage. Therefore we argue that we resolved the second *i** weakness reported in section 2.2, to embed the use of *i** in existing requirements practices in NATS and the HPA. No major usability problems were reported in both case studies. One possible explanation is that REDEPEND is built on the well-established Microsoft Visio application familiar to the analysts involved. Indeed the minimum levels of analyst training needed to use REDEPEND indicated a usable and effective toolkit.

The PiLGRIM method also provided the procedures and tool scale to model large socio-technical systems with *i** (*Claim-4*). The CAIT SR model contains just under 500 elements and links – the largest *i** model developed by us so far – and the HAP model contained over 200 elements and links. The upfront effort needed to generate large *i** models suggests that such models have a future role as reference models to be reused in multiple projects to analyse the impact of software requirements.

Finally the PiLGRIM method was useful to and usable by requirements analysts

(*Claim-5*). Analysis of the HAP change requests revealed 6 out of the 29 should be reconsidered before implementation to avoid possible errors or inefficiencies in the system. These errors or inefficiencies had not been detected using other analysis techniques in the HPA. One downside was the analysis expertise and effort needed – 5 person days to model a system analyse the impact of just 29 change request requirements. A more thorough cost benefit analysis is needed to support the claim fully.

9. Threats to Validity

Results were derived from data captured during two projects in industrial settings. Although this contributed to the external validity of the results and conclusions drawn from them, it also posed threats. This short section reports different conclusion, internal, external and construct validity threats [36] to the claims made from the two case studies:

Conclusion Validity

Threats to conclusion validity were concerned with issues that affected the ability to draw correct conclusions about the relations between the treatment and outcome. One threat is that our conclusions are drawn from just two case studies. The claims are therefore preliminary and should be interpreted carefully according to external validity threats reported below.

External Validity

The threats to the external validity were conditions that limited our ability to generalize the results from the two case studies more widely. Both case studies were similar – complex socio-technical systems with critical goals/soft goals and working practices that had evolved over time. Therefore our results have greater relevance for the effectiveness of PiLGRIM and REDEPEND in domains with similar characteristics and the budgets needed to support modeling and impact analyses, rather than for smaller systems and/or projects using more agile methods.

Internal Validity

Threats to the internal validity were influences that could have affected independent variables related to causality. The authors of PiLGRIM and REDEPEND played an active role in modeling in the first case study and trained analysts in both. Inevitably their participation and the desire of the businesses to make both projects successful had some influence over variables such as the size and sign-off of *i** models and number of impacts detected. That said, the scale and nature of both domains meant that their involvement had to be limited, and most analysis work with PiLGRIM had to be undertaken by the NATS and HPA analysts – the reported results were delivered and validated by these analysts.

Construct Validity

Construct validity concerned generalizing the results from the case studies to the concept or theory behind the study, namely the conceptual model in Figure 3. Results provide concrete evidence supporting the conceptual integration of *i** means-end links and satisfaction arguments in Figure 3 and the procedures and tool features built on top of this integration. This paper does not make wider claims about the integration of goal-based techniques and satisfaction arguments.

Clearly further development of PiLGRIM and REDEPEND and evaluation of their use in more requirements projects is needed to provide the evidence with which to

support or reject the claims reported in this paper. To make effective use of data that is collected an evaluation framework is needed to structure evaluation activities and relate data to claims.

10. Future Development of PiLGRIM and REDEPEND

Results reported in this paper can be used to make changes to PiLGRIM and REDEPEND. One is to support the concurrent development of *i** SR models and satisfaction arguments more effectively, similar to use of argumentation tools such as [1]. The low number of satisfaction arguments developed during the NATS case study provides evidence that the effort to produce them needs to be reduced. Another is to improve the usability of the requirements matrix in REDEPEND as both studies indicated that the effort needed to complete the matrix inhibited use. Specific features will enable analysts to:

- Refresh the matrix without losing already specified compliance values;
- Print the SR diagram overlaid with corresponding sections of the matrix;
- Reduce the effort needed to complete impact analyses within task and resource hierarchies based on automatic impact propagation to parent tasks;
- Analyse the impact of mutually-dependent requirements collectively.

We will improve the representation of satisfaction arguments in REDEPEND. In the current version, domain properties are associated with one end-element (requirement) and not the specific specification or refinement to which they relate. As such traceability is implicit, a physical method for such tracing within REDEPEND would improve usability and analyst understanding. This improved traceability will also enable us to extend the method to handle negative contribution links from specific means elements.

Another area we will improve is the support provided to the requirements analyst during the propagation procedure. For example, by following the existing propagation heuristics a goal or soft goal can be determined as non-compliant due to one of three factors – non-compliant means elements, an invalid argument and invalid domain properties. However, these are independent factors that should all be considered by the requirements analyst. Through tool support we will enable the analyst to record the reason(s) for non-compliance rather than just stating generic non-compliance. Also, where a domain property is determined invalid in one satisfaction argument, we will enable the requirements analyst to identify other satisfaction arguments containing this property which also may also lead to non-compliance. Both of these measures would improve the analyst's ability to undertake the propagation effectively.

Having improved the robustness of REDEPEND we will then apply it to model and analyse other socio-technical systems. Evidence from our two case studies suggested that the PiLGRIM method scales and is usable, therefore our final research direction is to investigate if our method provides wider benefits to an organisation. We look forward to reporting this research in the future.

Acknowledgements

The research reported in the first case study of this paper was supported by NATS, the UK's national air traffic service. The second case study was supported by the Health Protection Agency Centre for Infections.

References

1. Conklin J (2005) Dialogue mapping: building shared understanding of wicked problems. John Wiley & Sons

2. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Science of Computer Programming, Vol. 20, pp 3–50

3. Dick J (2005) Design traceability. In: IEEE Software 22(6), IEEE Computer Society, pp14–16

4. du Bois RM, Branthwaite MA, Mikhail JR, Batten JC (1981) Primary pneumocystis carinii and cytomegalovirus infection. Lancet. 1981 Dec 12;2(8259):1339. PMID: 6118728 [PubMed - indexed for MEDLINE]

5. Engmann J (2009) Evaluating the impact of evolving requirements on wider system goals: using i* methodology integrated with satisfaction arguments to evaluate the impact of changing requirements in HIV/AIDS monitoring systems in the UK. MSc Dissertation, School of Informatics, City University, January 2009

6. Guidelines Working Group (2001) Updated guidelines for evaluating public health surveillance systems. MMWR 50(RR13);1–35. Available at: http://www.cdc.gov/mmwr/preview/mmwrhtml/rr5013a1.htm [Accessed March 7, 2008]

7. Haley CB, Moffett JD, Laney R, Nuseibeh B (2005) Arguing security: validating security requirements using structured argumentation. In: Proceedings of the Third Symposium on Requirements Engineering for Information Security (SREIS'05) held in conjunction with the 13th International Requirements Engineering Conference, IEEE Computer Science Press

8. Haley CB, Laney R, Moffett JD, Nuseibeh B (2008) Security requirements engineering: a framework for representation and analysis. IEEE Transactions on Software Engineering 34(1), pp 133–153

9. Hammond J, Rawlings R, Hall A (2001) Will it work? In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering, IEEE Computer Society, pp 102–109

10. Horkoff J, Yu E, Lui L (2006) Analyzing trust in technology strategies. In: Proceedings of International Conference on Privacy, Security and Trust (PST 2006), pp 21–32

11. Horkoff J, Yu E (2010) Interactive Goal Model Analysis Applied - Systematic Procedures versus Ad hoc Analysis, The Practice of Enterprise Modeling, 3rd IFIP WG8.1 (PoEM10), pp 130–144

12. Horkoff J, Yu E (2010) Visualizations to support interactive goal model analysis, Requirements Engineering Visualization REV 2010 Fifth International Workshop on, IEEE, pp 1–10

13. Horkoff J, Yu Y, Yu E (2011) OpenOME: An Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool. In: CEUR Proceedings of the 5th International i* Workshop (iStar 2011), pp 154–156

14. Isaac AR, Doorn R (2006) Integrated cognitive analysis networks (ICAN): a tool to support cognitive task and error risk analyses in air traffic management. Unpublished document, NATS

15. i* wiki, 'Available i* Tools', http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Tools&structure=i%2A+Wiki+Home, (last accessed on 12th October 2011)

16. i* wiki, 'Comparing the i* Tools'http://istar.rwth-aachen.de/tiki-index.php?page=Comparing+the+i%2A+Tools, (last accessed on 12th October 2011)

17. Jackson M (1995) Software requirements and specifications. Addison-Wesley.

18. Jackson M (2001) Problem frames. Addison-Wesley.

19. Jones SV, Maiden NAM (2005) RESCUE: an integrated method for specifying requirements for complex socio-technical systems. In: Mate JL, Silva A (eds.): Requirements Engineering for Socio-Technical Systems, Ideas Group (2005) pp245–265

20. Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. ACM SIGSOFT Software Eng. Notes, 29(6), pp53–62

21. Lockerbie J, Maiden NAM (2006) Extending *i** modeling into requirements processes. In: Proceedings of 14th IEEE International Conference on Requirements Engineering, IEEE Computer Science Press, pp361–362

22. Maiden NAM, Jones SV, Manning S, Greenwood J, Renou L (2004) Model-driven requirements engineering: synchronising models in an air traffic management case study. In: Proceedings of CAiSE'2004. LNCS, vol 3084, pp 368–383. Springer, Berlin

23. Maiden NAM, Manning S, Jones S, Greenwood J (2005) Generating requirements from systems models using patterns: a case study. Requirements Engineering Journal 10(4), pp276–288

24. Maiden NAM, Lockerbie J, Randall D, Jones S, Bush D (2007) Using satisfaction arguments to enhance *i** modelling of an air traffic management system. In: Proceedings of 15th IEEE International Requirements Engineering Conference (RE'07), pp.49–52. IEEE Press, New York

25. Maiden NAM, Jones S, Ncube C, Lockerbie J (2011) Using *i** in requirements projects: some experiences and lessons. In: Social modeling for requirements engineering, ed Giorgini, Maiden, Mylopoulos, Yu. MIT Press

26. Mavin (2004) Scenarios in rail rolling stock with REVEAL. In: Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, ed Alexander, Maiden. John Wiley & Sons

27. Mylopoulos J, Chung L, Nixon B (1992) Representing and using non-functional requirements: a process-oriented approach. IEEE Transactions on Software Engineering 18(6), pp 483–497

28. NATS case study SR Model, available at: http://hcid.soi.city.ac.uk/research/NATS_RESCUE_SR_Model.pdf, (last accessed on 13th October 2011)

29. Ncube C, Lockerbie J, Maiden NAM (2007) Automatically generating requirements from *i** models: experiences with a complex airport operations system. In: Proceedings of 13th International Working Conference, REFSQ'2007, Trondheim Norway, LNCS 4542, pp33–47. Springer-Verlag

30. Organization Modelling Environment, http://www.cs.toronto.edu/km/ome/, (last accessed on 12th October 2011)

31. Perini A, Susi A (2011) Understanding the requirements of a decision support system for agriculture. An agent-oriented approach. In: Social modeling for requirements engineering, ed Giorgini, Maiden, Mylopoulos, Yu. MIT Press

32. Respect-IT (2007) A KAOS Tutorial, available at http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

33. Sutcliffe AG (2011) Analysing the effectiveness of socio-technical systems with *i** in Requirements Projects: some experiences and lessons. In: Social modeling for requirements engineering, ed Giorgini, Maiden, Mylopoulos, Yu. MIT Press

34. van Lamsweerde A (2000) Requirements engineering in the year 00: a research perspective. In: Proceedings of ICSE'2000: 22nd International Conference on Software Engineering, Invited Keynote Paper, pp5–19. ACM Press

35. van Lamsweerde A (2007) Engineering requirements for system reliability and security. In: Software System Reliability and Security, Broy M, Grunbauer J, Hoare CAR (eds.), NATO Security through Science Series - D: Information and Communication Security, Vol. 9. IOS Press, pp196–238.

36. Wohlin C, Runeson P, Host M, Ohlsson MC, Regnell B, Wesslen A (2000) Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Boston/Dordrecht/London.

37. Yu E, Mylopoulos JM (1994) Understanding "why" in software process modelling, analysis and design. In: Proceedings, 16th International Conference on Software Engineering, IEEE Computer Society Press, pp159–168

38. Yu E, Liu L, Li Y (2001) Modelling strategic actor relationships to support intellectual property management, 20th International Conference on Conceptual Modeling (ER-2001), Yokohama, Japan, November 27-30, 2001, LNCS 2224, pp164–178. Spring Verlag

39. Yu Y, Tun TT, Tedeschi A, Nunes Leal Franqueira V, Nuseibeh B. (2011) OpenArgue: Supporting Argumentation to Evolve Secure Software Systems. In: 19th IEEE International Requirements Engineering Conference, RE 2011, Trento, Italy, pp. 351-352. IEEE Computer Society

40. Zave P, Jackson M (1997) Four dark corners of requirements engineering. In: ACM Transactions on Software Engineering and Methodology 6(1), pp1–30