Argument Representation for Dependable Computer-Based Systems

C. GURR *University of Edinburgh*

Abstract: Society is becoming increasingly reliant upon the dependability of computer-based systems. Achieving and demonstrating the dependability of systems requires the construction and review of valid and coherent arguments. This paper discusses the need for a variety of classes of arguments in dependable systems and reviews existing approaches to the representation of arguments in each of these classes. The issues surrounding the certification of safety critical systems demonstrate the current need for richer representations of dependability arguments which support tools for their construction and review. The paper discusses how a meta-logical framework, informed by aspects of both formal and informal logic, offers a rich and unified means of representing dependability arguments and of thus addressing this need.

Résumé: La société dépend de plus en plus sur la fiabilité des systèmes qui emploient les ordinateurs. La réalisation et la démonstration de la fiabilité de ces systèmes exigent la construction et l'évaluation d'arguments cohérents et probants. Nous discutons du besoin d'avoir une variété de classes d'arguments dans des systèmes fiables et examinons des approches courantes pour représenter les arguments dans chacune de ces classes. Les préoccupations entourant la certification de la sécurité des systèmes critiques démontrent qu'il y a un besoin d'enrichir la représentation de la fiabilité des arguments qui appuient les outils employés pour construire et évaluer ces systèmes. Nous décrivons comment un encadrement méta-logique basé sur des aspects des logiques formelle et non formelle offre des moyens riches et unifiés de représenter cette fiabilité.

Keywords: argument representation, heterogeneous logic, computer-based systems, dependability, safety

I. Introduction

Society's dependence on *computer-based systems* continues to increase. By computer-based systems we mean any system (whether conceptual, physical, organisational or even social) in which some software component or components play a significant role. For the remainder of this paper we deal primarily with such systems. The systems themselves, embracing humans, computers and engineered systems, become ever more complex as they feed an insatiable appetite for new and extended functionality. Furthermore, these trends coincide with pressure for systems to be brought to market faster and at lower (and more predictable) cost. Achieving sufficient *dependability* in these systems, and demonstrating this achieve-

© *Informal Logic* Vol. 22, No. 3 (2002): pp. 293-321.

ment in a rigorous and convincing manner, is of crucial importance to the fabric of the modern Information Society.

The term "dependability", particularly when applied to computer-based systems, can be considered an umbrella term which encompasses, for example, availability, reliability, safety, and security issues. In general, a system may be deemed dependable if it reliably achieves the reasonable expectations which are placed upon it, and these expectations may combine aspects of availability, security and so forth. For example, for an on-line ordering system to justifiably be deemed dependable, we would expect that it should achieve a fairly consistent, high standard of availability (the system is not often "off-line", or "clogged up" with too many users), reliability (does not often crash, will carry out transactions accurately and faithfully), and security (does not permit unauthorised transactions, or unauthorised access to, or distribution of, personal or banking details).

Both achieving and demonstrating dependability in a given system generally requires the construction, negotiation, and assessment of valid and coherent *arguments* of dependability. These arguments are typically constructed and reviewed by broad and distributed teams who represent diverse interests and areas of expertise. Given the complex nature of the arguments which must be constructed, and the diverse audiences which must review them, the issue of *representing* dependability arguments is both an important and a complex one. In this paper we review the current state of the art in the representation of dependability arguments and find it regrettably lacking. Consequently, we focus upon a discussion of the need for, and requirements of, a richer representation of dependability arguments and of the lessons which Informal Logic has to offer in search of this goal.

In the following section we present a categorisation of the various classes of dependability arguments for computer-based systems, and review existing approaches for their representation. We examine in detail one illustrative class of argument—that of *certification arguments*, which are typically required to support claims that a system achieves some standard or measure of dependability. A particular example of this is in the domain of *safety critical* systems, systems whose failure may lead to loss of life or significant environmental impact. The developers of such systems are typically required to present arguments that their systems are dependable—that is, that the systems are acceptably safe. In Section 3, we review the issues surrounding the production and assessment of dependability arguments for safety-critical systems, as illustrative of the requirements upon dependability arguments in general. In Section 4 we discuss in detail the implications that such requirements have for the representation of dependability arguments—a representation that will need to combine elements of formal and informal logics, and will need to support both meta-logical reasoning and tools for the construction, negotiation and assessment of such arguments. Finally, in Section 5, we summarise and present the conclusions of these discussions, and indicate directions for future research.

2. Varieties of Dependability Arguments

For computer-based systems, we may distinguish between two varieties of arguments and argumentation for which dependability is an issue. On the one hand we have computer-based systems that *provide* arguments, or argumentation support, and for which the dependability of these arguments is a significant requirement. On the other hand we have computer-based systems for which dependability is a significant issue, and that therefore *require* arguments, or argumentation support, in assuring this dependability. We may make a further distinction in this latter class between arguments used to assist the design of dependable computer-based systems, most notably in exploring design options, and arguments used in the assessment of systems, particularly arguments which are required in support of some form of certification or as a regulatory requirement.

We may thus categorise dependability arguments into the following three classes:

Decision support: as used in "intelligent" systems which offer advisory support. Medical Informatics systems represent the classic example where the dependability of the advice offered is highly significant.

Design negotiation: arguments used in an exploratory fashion during the design of dependable systems. These arguments assist designers in the process of review and selection amongst a variety of design options.

Certification: (or *regulatory* arguments) such as those used to structure justifications that a critical, potentially hazardous, system is sufficiently safe and reliable. The structure of these arguments are typically informed by accepted standards or guidelines and the arguments themselves are reviewed by independent third parties.

In this section we review, for each of these classes in turn, existing approaches to the representation of dependability arguments.

2.1 Decision Support

Expert systems, an area of Artificial Intelligence research, has produced numerous "intelligent" decision support systems. For the majority of these systems, dependability—whilst desirable—is generally neither an absolute necessity nor a mandatory requirement. However, certain of these systems do operate in domains such as hazard analysis or healthcare, where the dependability, clarity and transparency of the advice being offered to support decision taking, is considered a significant priority. Such expert systems may be broadly classified as being of three main types: hazard analysis and avoidance; decision support; and monitoring and diagnostic systems.

The main focus of effort in producing decision support systems generally lies in the capture and codification of the expert knowledge that guides the advice that the system ultimately offers. Of relevance and interest to us here are those systems which, in addition to offering advice in support of some decision-taking task,

offer also the facility for review of the argument underlying the system's choice of what advice to offer. We do not seek to review expert systems at length here, but rather focus on illustrating an exemplar of one particular expert system which aims to offer dependable argumentation support for decision taking.

PRO*forma*, comprehensively described by Fox and Das (2000), is the combination of a formalism and suite of tools, developed at Cancer Research UK for the general purpose of building decision support and intelligent agents. PRO*forma* includes a language which is both a formal specification language (as that term is used in software engineering) and a knowledge representation language (as understood in AI), and also provides a set of software tools for building applications in the language.

PRO*forma* is, in essence, a first-order logic formalism extended to support decision making and plan execution, which it also incorporates a number of well known features of non-classical logics (e.g. modal logic, temporal logic) and two novel logics (LA, logic of argument and L OT , logic of obligation and time) to support decision making and action control. The suite of authoring and execution software that supports this formalism have, to date, been used to develop a number of clinical decision-support applications.

One example of a decision support system developed with the assistance of PRO*forma* is part of the RAGs project (Risk Assessment in Genetics), a collaboration between Cancer Research UK and a number of UK hospitals, which is designed to assist the general medical practitioner (GP) in assessing and communicating genetic risk information to women who are worried about their personal risk. The tool consists of a user interface which allows a family tree to be constructed for the presenting patient, incorporating information on the incidence of cancers. This interfaces with a risk calculation protocol written in PRO*forma* which generates an assessment of risk level and recommendations for patient management, along with an explanation for the conclusions reached. The user interface component is designed to be generic and may interface with different protocol software to assess the risk of different genetic conditions. The presentation of the argument for a particular decision is relatively straightforward: a list of risk indicators, as determined by the system, is presented together with a list of potentially mitigating factors—which may lessen the assessed level of risk for a particular case. The system provides a simple presentation, thus, of positive and negative factors as justification for some overall risk assessment level.

Expert systems for decision support, such as those developed using PRO*forma*, can often show remarkable sophistication in capturing and validating the expert knowledge which is required to guide the system. The design of the representation languages utilised by such systems can thus be similarly extremely sophisticated. However, this sophistication is largely applied in addressing the issues surrounding knowledge capture and representation and, as such, is less informative regarding the requirements for dependability arguments than are studies in the area of certi-

fication, as we shall see later.

2.2 Design Rationale

Work in the area of Design Rationale Capture has developed methods to capture and represent the arguments (rationale) which underpin the decision making process in design negotiation. Gruber (Lee 1993) defines design rationale as "an explanation that answers a question about why an artifact is designed as it is". Methods for representing design rationale thus typically focus on communicating the rationale *underlying* arguments rather than the argument itself, thus focusing on the justification of a particular argument or approach. The main advantages claimed of adopting such approaches are that they are useful as:

- a communication mechanism among design team to communicate past critical decisions, what alternatives were investigated, and the reason for the chosen alternative;
- a means of transferring design knowledge between projects with similar rationales;
- to encourage deliberation and explicit consideration of alternatives.

Existing approaches to representing design rationale range from relatively unstructured approaches—such as the use of electronic notebooks which utilise semi-formal approaches to capturing natural language arguments—to the use of requirements templates, and finally to entirely formal documentation of the rationale entities, their interdependencies, and so forth. Shum (1996) provides an excellent introduction and overview of work in this area, and earlier reviews may be found in Carroll and Moran (1991) and Moran and Carroll (1996). In general, following the classification provided by Lee and Lai (1991), approaches to design rationale fall into three categories:

Process-oriented design rationale: Historical records of design decisions, used during the actual design discussions;

Structure-oriented design rationale: Concerned with the structure of the space of all design alternatives, which may be constructed by post hoc consideration of the design process;

Psychological design rationale: Concerned with the human motivations underlying design choices.

The latter of these three categories is not of interest here. We next provide an overview of the seminal approaches in the former two categories.

Most work in process-oriented design rationale is based on Horst Rittel's, 1970s *Issue-Based Information System* (IBIS) (Rittell and Webber 1973), in which the design is documented as a hierarchical structure. IBIS provides a notation with 3 primitives: *issues*, which are questions that the design or argument is addressing; *positions*, which are potential resolutions of an issue; and *arguments*, that support or refute a position. Conklin (Conklin and Begeman 1988) subsequently produced

a graphical version called gIBIS, in which issues, positions and arguments are nodes in the directed graph, and the

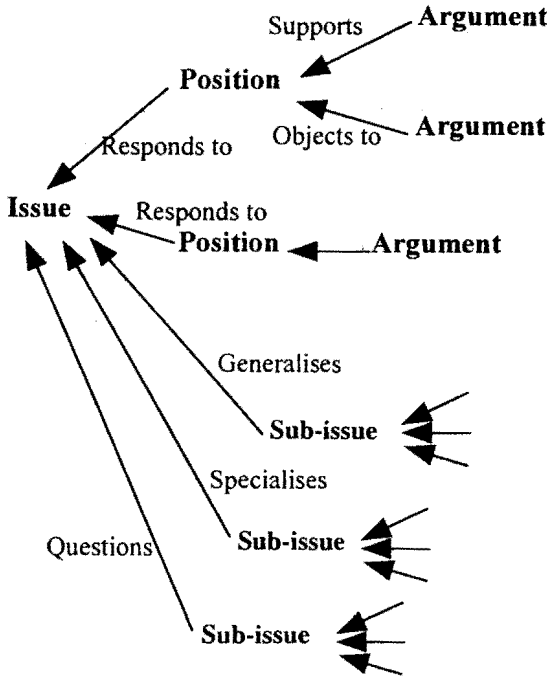

Figure 1: gIBIS: graphical *IBIS graphs*

connections between them are labelled to depict the relationship between connected nodes. As an example, Figure 1 presents the relationships between IBIS concepts as a gIBIS graph.

Structure-oriented design rationale, sometime also referred to as design space analysis, involves the post hoc reconstruction of the space of design alternatives and options that were considered during a project. Examples of systems developed to facilitate this process include: the *Questions, Options and Criteria* (QOC) notation, and *Decision Representation Language* (DRL). As with gIBIS, both QOC and DRL provide graph-based notations in which different types of nodes represent different basic elements of the notation, and directed links between nodes express relationships between them.

2.3 Certification Arguments

For many classes of dependable systems, both computer-based and otherwise, there is the possibility of constructing an argument to satisfy some, typically independent, third party that the system achieves some minimum specification or standard measure of dependability. That is, an argument that the system can be justifiably said to meet some specified level of—for example—security, reliability, quality or safety. A system developer's motivation for providing a dependability argument may be the desire to attain some internationally recognised quality standard for a business, a process or a product. Alternatively, as is often the case with safety- and security-related systems, there may be a mandatory regulatory requirement both to attain some specified standard, and to provide a dependability argument that argues the case for the attainment of that standard.

The domain of safety-critical systems provides an illustrative example of the regulatory requirements for, and of, dependability arguments. A safety-critical system is any system where a failure could potentially lead to significant human injury, or even loss of life, or to significant environmental damage. Examples of safety-critical systems come from a wide variety of industrial and other sectors. They include, for example, systems in the transport sector (road, rail, air and sea), from various energy industries (oil, gas, nuclear and electric), and numerous "hazardous" industrial sectors such as the chemical and petrochemical industries. The design, deployment, operation and decommissioning of systems from these, and other, safety-critical sectors is subject to both national and international assessment and regulation.

A significant part of the design process for safety-critical applications is the construction of a *Safety Case*, a collection of documents and data which together present the arguments for believing that the design of the proposed system is acceptably safe. The safety case sets out the risks involved with the operation of the process or equipment, and the possible consequences of failure. It also specifies what will be done (or has been done) to minimise the probability and the impact of these failures. The safety case is thus an example of a certification-style dependability argument, whose primary aim is to argue for the acceptable safety of some potentially hazardous system.

Given that a safety case is often a diverse collection of assorted forms of data, and that it is precisely the point of contact between the various parties involved in the production and assessment of safety critical systems, the management and communication of the information and arguments contained in a safety case is obviously an issue of primary importance. A number of commercial, software-based tools exist which purport to assist in addressing this issue. The majority of these tools are, in effect, document management tools which support the production and management of the various documentation that is required for safety cases and these tools are often specifically tailored to particular industrial domains. Of note as illustrative of tools which provide generic support for the production

and management of safety cases, and which specifically support the representation of the arguments which underly a safety case, are the *Safety Argument Management* (SAM) and *Adelard Safety Case Editor* (ASCE) tools.

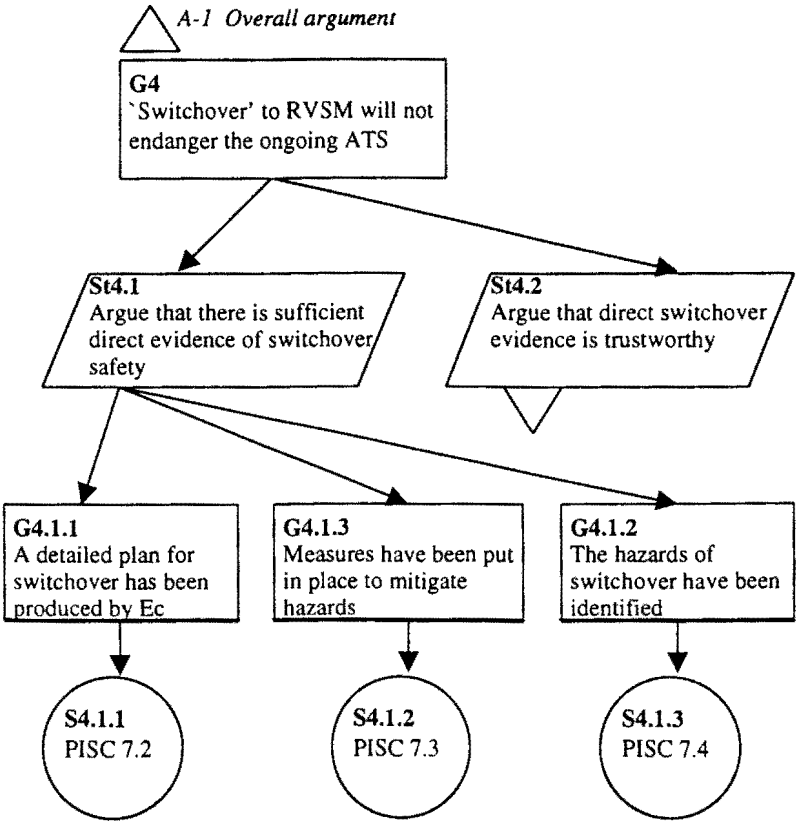

Figure 2: A part of EUR RVSM safety case argument

SAM [18] is a tool that supports the construction and presentation of safety cases and safety assessments, integrating documentation produced from a variety of safety assessment techniques. SAM utilises a graph-based notation, the *Goal Structuring Notation* (GSN) (Wilson, Kelly, and McDermid 1995), to represent the argument and evidence structure which underlies a safety case. GSN represents arguments as a graph with a variety of nodes which represent, variously: claims; supporting arguments; evidence; justifications and the like. Figure 2 illustrates the use of GSN to represent part of a safety case argument. This example is

an adapted version of a GSN diagram taken from the *EUR RVSM Pre-Implementation* safety case[1], which is required by the European Union as part of the assessment of *Reduced Vertical Separation Minimum* (RVSM), a planned change to Air Traffic Control practices across European airspace.

Figure 2 is a slight simplification of Figure A-17 from the EUR RVSM pre-implementation safety case[2], which presents the structure of the argument that "switchover" (the changeover period between current practice and the new RVSM) will be acceptably safe. This top-level claim is represented in Figure 2 by the box labelled G4, and is supported by two sub-arguments represented by the boxes labelled St4.1 and St4.2. The first of these sub-arguments is supported by a number of claims, represented by the boxes labelled G4.1.1 through G4.1.3 (in the actual EUR RVSM safety case there are, in fact, 5 sub-claims at this point rather than the 3 depicted here). The evidence which supports these claims, contained elsewhere in the safety case, is referred to by the circular nodes at the bottom of the graph. Finally, the triangular nodes associated with the boxes labelled G4 and St4.2 indicate that these nodes are, respectively, a sub-argument (of argument *A-1*) and have an associated sub-argument (in the case of St4.2).

ASCE[3] is a tool for building safety cases and showing the safety arguments they are founded upon. In common with SAM, ASCE goes beyond standard word processors—the usual tools for building safety cases—to support the development and presentation of a hypertextual safety case which shows the safety argument structure in a graphical format. These representations of arguments are similar to those of GSN, being graph-based representations with nodes of various types. ASCE differentiates between between material (nodes) acting as safety *claims*, *arguments* and *evidence*. It has a graphical tool for creating and presenting the safety argument structure and a further editor for editing the content of the nodes in the graphical network.

The GSN and ASCE notations share many similarities with the graph-based design rationale notations cited previously. However, the majority of these existing notations for representing dependability arguments provide little support for the *semantics* of arguments. That is, most of these notations offer little, if any, means of assessing the meaning of the elements of an argument, and thus offer little assistance in evaluating the quality, validity, strength, or similar properties, of any of the arguments they represent. In this paper we explore the requirements of a richer language for representing dependability arguments, a language in which issues of the meaning, validity and quality of an argument are captured and may thus be examined and reasoned over. To motivate and illustrate these requirements, and to guide the form of such an argument representation language, in the following section we explore in more detail the role of arguments and argumentation in the construction and review of an exemplar of dependability arguments: the safety case.

3. Argumentation and the Safety Case

A safety case is collection of documents and evidence, produced and collated by the developers of a safety-critical system, which is primarily aimed at convincing the regulatory authorities that the proposed system is acceptably safe. The safety case also has an important secondary role as a repository of analyses of the system and documentation of safety procedures, failsafe mechanisms and the like. This paper focuses on the primary role of safety cases as *arguments* of system safety.

The safety case is a point of contact between experts from disciplines as diverse as electronic engineering, town planning, nuclear science and medicine. It contains contributions from a broad and distributed team of designers, suppliers and analysts. For large and complex systems, the production of the safety case may be the responsibility of the procurers or end users, who must therefore collate, organise and present evidence and arguments for safety supplied from both within their own organisation and from external suppliers. The design and assessment of safety critical systems often involves broad and distributed teams of designers, suppliers, users and analysts who represent diverse areas of expertise and motivations and a broad range of technical and non-technical disciplines. Safety cases are generally assessed by independent organisations who must ensure both that the arguments for safety are valid, and that relevant industrial standards and guidelines have been adhered to throughout.

The growing use of programmable devices is making system safety assessment increasingly complex, at the same time rendering many standard safety engineering techniques inapplicable. More and more frequently, safety cases have to combine the uncertainty of statistics with the abstract absoluteness of logics, and assess the validity of formal proofs in the context of unexpected and asynchronous events. Software can be considered to be safety critical only in the context of the larger system: of the hardware in which it is implemented, the attached peripherals and the operating environment. An error in software cannot directly cause injury, but the consequences of a software error upon the attached peripherals or operating environment—that it causes a missile to be launched unintentionally, prevents the brakes of an aircraft from operating when it lands, or fails to correctly shut down a nuclear reactor in an emergency situation—may cause injury, fatality or major environmental damage. When considering the safety, correctness and dependability of safety critical software we must therefore consider the software in the context of the whole system and the environment in which it operates. This entails drawing upon a body of expertise from domains as diverse as electrical, chemical, automotive, aero-spatial and nuclear engineering; risk assessment; and environmental analysis. The safety case is thus a collection of data drawn from multiple sources, representing diverse disciplines, which must be assessed by a range of both technical and non-technical experts, who exhibit a range of differing interests. The accurate and effective communication between these groups

of the structure of dependability arguments is therefore an issue of primary importance.

Safety cases typically contain a preamble, defining context such as environment(s) of expected use and lifetime; the conclusions of a hazard analysis, identifying likely hazards and their severity; an argument as to how the specification will overcome or mitigate these hazards to an acceptable level of safety; an argument as to how the implementation can be assured to comply with the specification; and an argument that sufficient procedures and safeguards are in place to ensure that operation, maintenance and eventual decommissioning of the system will similarly assure acceptable safety.

The two main protagonists in the construction and review of safety cases are the developers of the proposed system and the assessors, who are members or representatives of the relevant regulatory authorities. In certain industrial domains a safety case may be assessed by more than one regulatory authority. An example of this is the UK's rail transportation network, which is currently overseen by a number of more-or-less independent authorities, including the HSE (UK government's "Health and Safety Executive"), Railtrack (responsibility for tracks and signalling), the SRA ("Strategic Rail Authority", a quasi-independent policy institute which was initially created as a subsidiary of Railtrack), and a number of train operating companies—profit-making businesses between whom the UK rail network has been split into geographic regions. Naturally, producing safety cases to simultaneously satisfy such a complex collection of organisations with varying authorities and responsibilities, is no trivial matter. Byzantine regulatory environments aside, however, the process of reviewing safety cases is typically one of negotiation between developers and assessors. It is common for the process of review to undergo several phases where the developers are given the opportunity to respond to issues raised by the assessors, being encouraged to revise and resubmit a safety case that was failed in an earlier phase. Ultimately, however, responsibility and authority for the acceptance or rejection of a safety case rests with the assessors.

A significant difference between safety critical systems and non-safety-critical systems is in the degree of attention paid to assessment and testing, often enforced by industrial standards, both of the design and the design process of such systems, and of the system in use. These standards often enforce mandatory requirements upon the design, development, testing, assessment and certification of such systems. For safety critical systems in which software plays a significant role a newly emerging standard of particular import is IEC 61508[4], which addresses the functional safety of such systems. Both the form and content of safety cases are significantly informed by standards such as IEC 61508, as it is partly according to compliance with the stipulations of these standards that safety cases are judged. There are a variety of standards, both generic and specific to various industrial domains, both national and international, and both providing suggested guidelines

and stipulating mandatory procedures and policies. In the past decade or so there has been a notable change in standards, moving away from the *prescriptive* standards of the past towards more *goal-setting* standards that are common today. This change has placed greater responsibility on the developers of a safety case to present a convincing argument that there is justifiable confidence in the safety of the proposed system. In the following section we review the motivations underlying the move to goal-setting standards such as IEC 61508, and discuss the effect this has upon the structure, form and content of safety cases.

3.1 Goal setting standards

The domain of safety critical systems is one that can be said, albeit with a degree of (justifiable) cynicism, to be *disaster-driven*. That is, that standards, regulations and requirements will change over time, and more often than not, the driving forces that motivate these changes are significant disasters or catastrophic incidents. A notable example which resulted in a major change for the philosophy underlying safety arguments across all industrial domains, in Europe at least, was the Piper Alpha disaster of 1988.

On 6 July 1988, on the offshore oil drilling platform Piper Alpha, 100 miles off the East Coast of Scotland in the North Sea, there was an escape of flammable gas. The gas ignited causing an explosion which led to large oil fires. The heat ruptured the riser of a gas pipeline from another installation, producing a further massive explosion and fireball that engulfed Piper Alpha. All this took just 22 minutes. The scale of the disaster was enormous. 167 people died, 62 people survived.

The extent of death and injury resulting from the Piper Alpha disaster was of great concern to the offshore petroleum industry worldwide, and widespread reviews for safety equipment and emergency response were undertaken. In the UK a public enquiry, undertaken by Lord Cullen, was commissioned in July 1988 to establish the circumstances of the accident and to make recommendations as to the future safety regime. His report led to extensive restructuring of the UK offshore safety legislation, with the primary onus of responsibility for offshore safety being shifted towards the operating companies and away from the regulatory authorities.

The Cullen inquiry confirmed that that the owners of the Piper Alpha platform had complied with relevant safety regulations. In particular, the inquiry confirmed that various emergency life-saving measures, such as the provision of an emergency room—fire-proof "bolt-hole" for workers, had been included in the platform as prescribed by regulations. However, this emergency room was, in actuality, virtually inaccessible during emergency situations such as that of 6 July 1988 and, as such, did not succeed in keeping the Piper Alpha workers safe. The regulations in force before July 1988 stipulated that an emergency room was required for offshore oil platforms, but compliance these stipulations did not, clearly, guar-

antee that this emergency room would actually be able to serve its intended purpose.

Lord Cullen specified that *goal-setting* regulations, which require certain objectives to be met using appropriate methods, be implemented on offshore oil platforms to replace the formerly *prescriptive* regulations, which imposed detailed measures that had to be taken invariably. Non-mandatory guidance notes would accompany goal-setting regulations to facilitate meeting goals effectively. Goal-setting regulations, though they may seem like a subtle change in procedure, actually alter safety methods considerably.

Under prescriptive regulations the operating companies were required simply to produce evidence that they had complied with the mandated measures, and any consideration of whether this actually achieved acceptable safety was the responsibility of the regulators. Lord Cullen was highly critical of this approach as it could, and in the case of Piper Alpha clearly had, foster a situation in which the operating companies merely complied with "the letter of the law" without regard as to whether compliance actually achieved acceptable safety (the "spirit" of the law). The move to goal-setting regulations shifted the burden of proof onto the operating companies, who now were required to *argue*, in the safety case, that their proposals would achieve acceptable safety. While this move to goal-setting regulations was initially directed at the offshore petroleum industry, the implications of the lessons learned from Piper Alpha were felt to be clearly applicable to the safety-critical industries in general and over the past decade goal-setting regulations, embodied in standards, have become the norm for all safety-critical domains. Consequently the need for developers and operators of safety-critical systems to construct and present convincing arguments of acceptable safety, contained within their safety cases, has become a major aspect of their business.

The safety case constructs an argument that the system is *acceptably safe*. What it means for a system to be acceptably safe is itself open to some interpretation, with the commonly accepted definitions again typically set out in standards. In general, these definitions assert that for a system to be "acceptably safe", the likelihood of it being involved in some event with severe consequences (such as loss of life or significant environmental damage) is, for its expected operating lifespan, sufficiently low. A specific definition of "acceptably safe" will typically assign numeric values, such as, that the probability of any undesirable event is below some stated value; that the probability of an undesirable event over some given period of time is below some stated value; that the probability of any single demand upon the system is below some stated value; or that the estimated mean time between undesirable events is above some stated value. The details of any particular definition are largely determined by the type of system and by the industrial domain in which it operates. In the main body of the safety case is presented the argument that the proposed system achieves the level of safety set out by this definition and the evidence in support of this argument. This evidence can include

such diverse elements as test data, analyses of formal models of the system, appeals to expert judgement and evidence of compliance with "industrial best practice"—generally understood to be determined by standards and potentially demonstrated through independent certification. Furthermore, a common expectation of the designers of safety critical systems is that they will incorporate hazard mitigation features or procedures into the design—that is, safeguards and procedures designed to reduce the detrimental impact of various hazards if they actually do occur—and the presence and claimed effectiveness of these mitigations will also, typically, be described in the safety case.

3.2 The structure of safety cases

A safety case is usually prefaced by a description of the *context* in which the subsequent argument of system safety is situated. This can include a description of the environment or environments in which the proposed system is expected to operate, and of the expected lifespan of the system. Often the preface will include a list, assumed not to be exhaustive, of the significant hazards or risks that the system might face in these environments. The construction of such a list of hazards is itself a regulated and reviewable process, for which there are widely accepted and standardised analytical techniques for the identification, quantification and prioritisation of potential hazards.

We may again look to standards as the primary indicators of the structuring of the evidence and arguments contained in a typical safety case. For the European Union, one of the most important industry-generic standards is the recent IEC 61508. This standard accepts that most safety cases will commence with a description of the context of the proposed system, and insists that this is followed by a thorough hazard analysis. The conclusions of this analysis set out the systems' risks: the unsafe consequences of the system (loss of life, detrimental environmental impact), and the potential manners by which these unsafe situations might conceivably arise. From this description of the risks of the system is derived, and recorded in the safety case, a set of *safety requirements*, which in effect detail the constraints and safeguards which must be enforced upon the system to ensure that the identified risks do not arise. These safety requirements inform the bulk of the remaining safety case, as ensuring the requirements are met is equated with achieving acceptable safety.

In common with many earlier standards, IEC 61508 presents a fairly general model of a safety critical system's "lifecycle". The main stages of this lifecycle are: the systems conception; its specification; the implementation of this specification; the operation of the system; its maintenance; and its eventual decommissioning. For each of these lifecycle stages, IEC 61508 sets out the various forms of analyses, documentation (of, for example, operating and maintenance procedures—both routine and emergency) and evidence that should be produced and recorded in the safety case. The safety case presents this information in support of argu-

ments that, for each lifecycle stage, the safety requirements are met and thus that acceptable safety will be achieved.

IEC 61508 suggests specific analytical tools and techniques for each of the various lifecycle stages. However, while the standard insists that *some* analysis is carried out at each stage, it does not insist that developers must use the suggested techniques. Developers are free to use alternative techniques as long as they present arguments to mitigate these deviations from the standards. These arguments must convince the assessors that the alternative techniques used by the developers are at least as effective as the IEC 61508 suggested techniques that they have replaced.

The analyses and evidence that must be produced, as recommended by IEC 61508 and similar standards, are many and varied. They include formal (mathematically-based) modelling, testing, probabilistic risk assessments, and appeals to external (to the developer) evidence and arguments—such as expert judgement, or the use of system components which are certified by accepted independent authorities. Furthermore, this wealth of information in the safety case combines evidence and the results of analyses constructed from a wide and diverse variety of areas of expertise, potentially including the results of architectural, environmental, mechanical, electrical, structural and software studies in addition to those of whatever industrial domain the system operates in, be it chemical, petrochemical, energy (nuclear, gas, electricity, or other), transport (road, rail, air or sea), or military.

The combination of this often vast quantity of disparate data into safety case arguments which must be accessible, coherent and convincing to those charged with assessing them, is clearly a significant concern for system developers. Thus, it is easy to understand the strong desire, on the part of both developers and assessors, for tools and techniques to assist in the construction and review of safety case arguments.

3.3 The form of safety case arguments

The first step in constructing a safety case is the hazard analysis. The methods and tools of hazard analysis are mature and well understood, and standards can afford to tend more to the prescriptive in their demands for what must be carried out. For the engineering of software, as a less mature discipline, hazard analysis techniques are less well developed and the subject of ongoing research. However, as hazards are typically physical in nature, the hazard analysis of software specifically is not an issue we need consider here. Furthermore, there is generally little argumentation in the hazard analysis and so, again, our interest here is primarily in the post-hazard analysis stages of safety case construction and review.

A study of safety-cases from a sociological perspective by MacKenzie (1996) suggests that, essentially, computer-system safety cases draw upon four different kinds of argument:

induction: this system is safe because testing shows that it is safe;

deduction: this system is safe because it can be proven mathematically that its design is a correct implementation of its specification, which is safe

construction: this system is safe because the process used to design it is a safe process that leads to safe systems;

authority: the system is safe because an expert authority has judged it to be safe

Two of these, deduction and induction, are most often utilised in the analysis of specifications and implementations respectively. Software, however, differs noticeably from traditional physical engineered systems in that its modes of failure are less predictable and more likely to happen "without warning". Consider, for example, a load-carrying beam which tests have shown can carry a given weight. We may clearly assume that the beam can also carry any load of less than this weight. Over time, the beam may deteriorate so that it will ultimately fail, but there are likely to be observable signs of immanent failure so that this too is—to some extent at least—predictable. If a piece of software, by contrast, is shown to behave correctly for a given input, we may still make absolutely no assumptions about its behaviour for different, even subtly different, input. Furthermore, when software does fail to behave as intended for some particular input, there are seldom any observable indications to this failure prior to its occurrence. Thus, while tests of physical components permit us to make inferences about their behaviour in a broad range of situations, tests of software components—which are often far more complex functionally than their physical counterparts do not typically permit us to make similar inferences of general behaviour. This difficulty in evaluating the software *product* through testing, is often compensated for by evaluating rather the *process* by which the software was developed. Thus we have a third type of argument utilised in safety cases for software-based components, named argument by construction by MacKenzie.

The fourth type of argument identified by MacKenzie is that of authority. The interpretation of this term is equivalent to argument from cognitive authority, or appeal to expert judgement. However, other forms of arguments by appeal to authority are prevalent throughout safety case arguments and in fact will typically underpin or justify particular inductive, deductive and constructive arguments. Inductive arguments, based upon testing, may use appeals to administrative or cognitive authorities to justify both the choice of a particular testing strategy and the conclusions drawn form the testing. Constructive arguments may similarly use appeals to authority to justify the acceptability and worth of the chosen development process. In typical deductive arguments, some model of the system or a component is proven to possess desirable qualities or to prevent undesirable situations. These proofs may be mathematically incontrovertible, however the ques-

tion remains as to whether the model used is a sufficiently accurate representation of the reality of the implemented system or component. Again, an argument from authority—typically an appeal to the expert judgement of the creator(s) of the model—will be the underlying justification of the validity of this deductive argument.

An important issue in the representation of safety case arguments is the ability to assess the *strength* of any particular argument. We would wish to be able, with reasonable objectivity, to assess both whether a given argument is sufficiently convincing and, if convincing, whether it hides any weaknesses that should be of concern. Studies in Informal Logic are of enormous value here, as they offer a rich understanding of what it means for an argument to be plausible or convincing, and of the circumstances under which steps in an argument that are not strictly deductively valid—such as appeals to authority—may be determined to be reasonable.

Walton (1989) discusses appeals to authority at some length, defining different varieties of appeals to authority, common errors in certain types of appeal, and sets of "critical questions" which must be asked of any such appeals to assess their worth. Walton divides appeals to authority into two varieties which are very different in nature (although noting that any given appeal may possess elements of both. Authority is an ambiguous term which may refer either to an *administrative* authority, one who has the right to exercise command in some relevant situation, or to a *cognitive* authority. That is, one who has expertise in some domain of knowledge. An appeal may be made to either of these forms of authority. The citing of expert opinion is a common example of the latter.

Following Walton, we may describe appeals to expert authority by the following schema:

E is an expert in domain (of knowledge) D

E asserts that A is known to be true

A is within D

Therefore, A may (plausibly) be taken to be true.

With this schema, and the sets of common errors and critical questions discussed in Walton (1989), we may see how any given appeal to expert authority should be examined for its plausibility. Suppose, for example, that expert E is cited as asserting that statement A is known to be true. Many questions must be asked of this appeal before we may assert, with reasonable confidence, that we believe A to be true. We must ask, for example, is it true that E is known to be an expert in domain D and, if so, is it true that A is knowledge that is actually within that domain? Furthermore, we must ask whether E* has actually made the assertion A, or whether some interpretation of their statement has taken place. If this is so, then further questions must be asked to determine whether that interpretation is both accurate and reasonable. Finally, we must assure ourselves that the cited pronouncement has been, or can be, validated with a reasonable degree of confi-

dence. Is there, for instance, objective evidence in support of the pronouncement? Is there, perhaps, disagreement on this topic between the cited experts and other authorities, and do these disagreements bring the pronouncement into question? A representation of safety case arguments must permit such questions to be asked. Knowledge gained from the study of informal logic, such as the understanding of what is required to validate appeals to authority, we may learn much concerning what is required to assess safety case arguments, and thus about both what is necessary and what is possible in the representation of safety case arguments.

Further variants of appeals to authority and other extra-logical forms of justification may be observed in safety cases as warrants for both the stating of assertions and in arguing for confidence in particular assertions. One common example of a further variant of an appeal to authority is a claim that some aspect of a system is acceptably safe because industrial "best practices" have been followed. Such an appeal may be used to justify the use of a construction argument, for example. Alternatively an appeal to best practice may be used in support of an induction argument, as justification for the adoption of some particular testing strategy or method. The phrase "best practices" can, and has, been applied to many aspects of the development and operation of safety-critical systems. For example, design processes, analysis techniques, design philosophies, operating procedures and emergency mitigation procedures can all be asserted as following accepted industrial best practice. The knowledge as to what constitutes best practice can be embodied in standards or can be viewed as being widely accepted. One example of this, both in standards and commonly accepted, is the use of *redundancy* to improve confidence in safety.

There are two common forms of redundancy used in safety-critical systems. The first is the use of redundant components in systems, improving dependability through the existence of either "back-up" or "checking" components. One variety of this form of redundancy which is accepted as being best practice is *Triple Modular Redundancy* (TMR): the use of three equivalent components in parallel for some critical function. The advantage of TMR over double redundancy is seen in situations where a component, rather than simply failing, can fail in opposition to its intention. For example, a cable intended to pull an aircraft aileron into the up position could fail by pulling the aileron downwards. In such a situation the two "healthy" components will "outvote" the failing component, thus ensuring correct behaviour. The use of TMR in critical components is generally accepted to be industry best practice. It should however, be noted that the application of redundancy, including TMR, to software components is a difficult and open issue, as identical software components typically exhibit identical behaviour, including identical failures, and thus redundancy will not improve reliability.

The second common form of redundancy for safety critical systems is in arguments of system safety. It has been known, and is advised by certain standards, for developers to construct "multi-legged" safety arguments. That is, argu-

ments consisting of two or more redundant "legs", each of which is by itself an argument of system safety. The reasoning here is that these multiple, redundant, arguments support confidence in each other, as potential errors in any one leg (mistaken assumptions, flawed inferences, misinterpretation of data, and so forth) are mitigated by the existence of the other, independent argument legs. An example of such multi-legged arguments is the safety case constructed to justify the protection system for the UK's Sizewell B nuclear reactor. Sizewell B was the first UK nuclear reactor to implement an entirely software-controlled emergency shutdown protection system, and the safety case justifying the use of this, for that environment, novel technology was two legged, consisting of (allegedly) independent process- and product-based arguments.

A representation of safety case arguments must accommodate each of the forms of arguments described here: induction, deduction, construction, appeals to authority and other non-logical warrants. Furthermore, such a representation must support the *combination* of sub-arguments of all of these forms, and permit the objective evaluation of such complex, combined arguments. We next clarify the process by which safety case arguments are typically evaluated.

3.4 The process by which safety cases are negotiated

Walton (1989) describes argument dialogues as consisting of four clear, if not necessarily distinct, stages: opening; confrontation; argument; and closing. In the opening stage the type of argument is made clear and the dialogue rules which will govern the dialogue are agreed. The choice of dialogue rules will determine: specifics of the language to be used; the rules for turn taking in the dialogue; strategic rules (what constitutes a win or loss); and rules of commitment—under what conditions propositions are accepted and when responsibility for a proposition is entailed. In the confrontation stage the issue under discussion is announced, thus clarifying the goal or goals of the argument. The third stage is the argument itself, and in effect continues until both parties accept either a win or a loss (a third possibility, of *stalemate*, is of course possible, but does not concern us here). Once the goal is fulfilled, the dialogue enters the final, closing stage, where various agreements, consequences or future actions and responsibilities may be ascribed to the participants.

Of a variety of types of argument studied in informal logic, the negotiation of safety cases is best understood as a form of *persuasion dialogue* (Walton 1984), also known as a "critical discussion". The negotiation of safety cases is in fact closely related to that of an inquiry—a subcase of persuasion dialogue. Informal logic has studied one type of inquiry, the legal argument, at great length. However, the safety case negotiation differs from the legal argument in two key respects. Firstly that, while assessors will subject a safety case to rigorous and exhausting scrutiny, the negotiations between developers and assessors are largely cooperative and very rarely display the confrontational nature that often characterises legal

negotiations. Secondly, while safety case negotiations will conform to accepted dialogue rules and forms, these are nowhere near as rigid, formulaic or as firmly established as those in legal argumentation.

In safety case negotiations, the first two of the four argument dialogue stages (the opening and confrontation stages) are pretty much standardised and thus fairly perfunctory. The issue under discussion (whether the proposed system is acceptably safe) is generally both clear and fixed, the dialogue rules are widely known and accepted, and the goal and strategic rules for determining whether it is achieved (a win being ultimate acceptance of some variant of safety case as legitimate justification for approval of some variant of the proposed system, a lose being ultimate rejection of the same) are likewise generally understood. The fourth and final argument stage, the closing, is similarly generally understood and accepted—there often may be many and substantial further requirements and obligations placed upon the developers, but these are not of interest here. Of interest to us is the third stage, the argumentation stage.

In the argumentation stage the assessors have the right to investigate and question all evidence and arguments presented in the developers' safety case. The assessors may freely, as they see fit, qualify presented evidence, question assumptions, question or reject appeals to authority, and refute inferences made. The developers may respond to any such issues raised by the assessors by strengthening their safety case. This strengthening may include, for example, adding extra evidence, further justifications of assumptions or of appeals to authority, further such appeals, or even the offering of alternative arguments to those in the original safety case. The developers may also alter the proposals regarding the system that the safety case supports. They may, for example, qualify claims about the context of the system (its potential operating environments and lifespan), or they may offer to alter the design, either by including additional safeguards or even by making alterations to the design itself. This negotiation process—in which the safety case may be critiqued, amended, and critiqued once more—continues until the assessors determine that they have reached a final judgement of whether to accept or reject the safety case. In the case of ultimate rejection, further appeals by the developers are possible and may well bring in legal aspects to an ongoing argument dialogue, but for our purposes these do not differ significantly from continued negotiations over the safety case.

Given the situation we have presented here—where safety cases combine complex and diverse evidence into arguments of acceptable safety, for review by, potentially, a number of regulatory authorities through a process of (generally cooperative) negotiation—the desire for an effective means of representing safety cases arguments is apparent. Such a representation must accommodate both logical and extra-logical warrants, must permit the combination of quantitative and qualitative assessments of safety—of the results of formal, mathematical analyses, empirical assessments and appeals both to a variety of forms of authority and

to a variety of further, extra-logical warrants. Furthermore, such a representation must support both developers in the construction of safety case arguments and assessors in their review, and must support both parties in the process of negotiation and revision of safety cases. In the next section we discuss in more detail the requirements for representations of safety case argument representation, and present our proposals for a language of such representations.

4. A meta-framework for argument representation

There are many benefits to having a formal representation of argumentation. A clear syntax permits us to be precise about both the objects that the representation deals with and the kinds of statements we may make—the sentences in the language—concerning these objects. A clear semantics permits us to be precise about the meaning of these statements and, given both a model-theory and a proof-theory, we may also be precise about what constitutes a *valid* argument in this representation. However, any representation of dependability arguments must also accommodate extra-logical forms of reasoning over and above those representable in classical formal logics, such as first order logic. A representation of dependability arguments must support not only logical arguments but also numerical (probabilistic) ones, temporal arguments (where evidence has a finite lifespan), judgements of confidence, and other such informally warranted arguments.

We assert that classical logics, whilst a useful starting point, are clearly insufficient for the representation of dependability arguments. In this section we first enumerate the various forms of arguments over and above those permissible in classical logic that a representation of dependability arguments must accommodate. We further argue that, rather than simply a first-order representation of dependability arguments, what is required is in fact a *meta-level framework* in which we may not merely represent arguments (as first order theories) but may also reason about these arguments themselves. We discuss the many issues that such an approach raises, primarily:

- what features and attributes are required to represent dependability arguments as first order theories?
- what forms of reasoning over, and analysis of, dependability arguments are required—and what does this suggest in terms of a meta-framework for representing and manipulating them?
- what is necessary to integrate this framework with existing and future approaches to argument representation, analysis, reuse and tool support?

Following discussion of these issues, we present a proposal for such a meta-framework for the representation of dependability arguments.

4.1 Requirements for the representation of dependability arguments

We may take *evidence* to be the atomic elements from which dependability arguments are constructed. To represent such arguments we thus require a language whose atoms are citations of (bodies of) evidence. Arguments may then be constructed from these atomic elements just as more complex logical sentences are constructed from atomic propositions by using logical operators such as conjunction, disjunction, negation and so forth. However, in contrast to logical propositions, the evidence cited in dependability arguments often possess numerous features not captured in classical first-order logic. An obvious example being quantitative measures, such as probabilities or assessments of confidence in the evidence. Furthermore, in dependability arguments the evidence presented may be both cited and composed in many ways beyond those understandable as classical logical operations.

Quantitative values are often associated with the evidence presented in dependability cases, particularly when the evidence cited refers to the results of the testing of some component or subsystem of the system under consideration. Such, typically probabilistic, measures must be combined and it can be a requirement of the argument that it present the probability associated with the overall argument, a value calculated through composition of the values associated with the individual items of cited evidence. Safety cases, for example, are often required to present an overall measure of the probability of unsafe behaviour, and to argue that this probability is sufficiently low. Further quantitative values may also be associated with particular pieces of evidence. For example, appeals to expert judgement can have associated with them some measure of confidence in the cited assertions. This measure may be either, or both, of confidence by the appealer in the expert or of confidence by the expert in their assertion. Similarly, in the fragment of the EUR RVSM safety case argument shown in Figure 2, we see that the collection of evidence supported the claim that switchover is safe (Claim St4.1), is associated with a "confidence" argument—an argument that the cited evidence is trustworthy (Claim St4.2).

Evidence in a dependability case may have other associated features in addition to quantitative values. Another common example being that evidence often has an associated *vitality*. That is, any cited evidence typically carries with it some constraints over the period of time for which it may be considered valid. It may be that, at some point in the future, a particular piece of evidence is no longer considered valid. This point in time may be known at the point at which a dependability argument is constructed, or it might be that in the future certain classes of evidence are retrospectively asserted to have become invalid. It has been known, for example, for new or updated industrial standards to mandate changes in assessment that can invalidate evidence or even arguments derived in earlier safety cases from forms of analysis that are now deemed to be no longer trustworthy or reliable. This then requires the re-appraisal of safety cases which cite evidence or

arguments that are derived from these, now outdated, forms of analysis. Alternatively, it is conceivable that one's measure of confidence in a piece of evidence, say the results of a some analysis of a specification, may diminish over time as the evolving reality of the implemented system "drifts" further and further away from the idealisation of the original specification.

To capture such quantitative and qualitative attributes in the representation of dependability arguments we may associate features with the basic atomic elements of the representation. In certain cases these features may be very simple in nature, such as simple binary values, whereas in other cases – such as in the use of temporal features – we will likely require a language for expressing the necessary details of the feature associated with any particular piece of evidence. For example, to represent the vitality of a variety of evidence we may choose to use a feature-language with the expressiveness of a complete propositional, temporal logic. There are several logical formalisms that associate features with their primary objects of discourse. The majority of these are subsumed under the term *description logics* [3] and their most important domains of study to date have been in the areas of knowledge representation and the semantic analysis of natural language. These studies, particularly in the area of knowledge representation, offer the foundations for a variety of feature-languages.

In addition to associating features with evidence, we would wish to be able to identify, and refer to, particular claims or items of evidence. For identification and reference purposes a simple set of distinct labels would suffice. However, the use of more sophisticated languages of labels to identify both atomic propositions and individual steps in a logical proof, as explored by Gabbay's *Labelled Deductive Systems* [9], would permit both modularisation of arguments (a sub-argument is referenced via the label of its top-level claim) and the use of non-classical, context sensitive, operators—as in, for example, varieties of Linear Logic [10] where an item of evidence may be used at most once in an argument.

The primary element of an object language for representing dependability arguments is thus a *claim*. Claims have features, or attributes, and are identified by unique labels. Basic claims are taken to be particular pieces of evidence, and complex claims are built up from these using logical and extra-logical *warrants*. A dependability argument is thus a proof in this object language, supporting the top-level claim. While evidence and claims may have many features or attributes associated with them that are not typically represented in classical logic, the means by which evidence is composed to construct complex dependability statements also goes beyond the forms of compositions permitted by classical logical operators. Many extra-logical warrants are permissible in dependability arguments, such as various forms of appeals to authority, "best practice" and "multi-legged" arguments. A language for the representation of dependability arguments must also accommodate such extra-logical warrants. In practice, to support the representation of arguments that are in the process of construction, it is likely that certain

claims will be flagged as being "incomplete", or "as-yet-unvalidated", and these extra attributes and their consequences must also be accommodated as extra-logical warrants for the acceptance of incomplete claims.

To make these various accommodations, such extra-logical warrants will take the form of extra operators, similar in character to the logical operators such as classical conjunction, disjunction, implication, negation and so forth. As extra operators, these extra-logical warrants will thus claim their own rules in the proof theory and semantic interpretations in the model theory.

Finally, as arguments are modular, we may construct *heterogeneous* arguments, where distinct legs of the argument utilise distinct local logics. Thus, certain sub-arguments could utilise relatively sophisticated probabilistic or temporal feature-languages as required, with other sub-arguments utilising linear or other non-classical logics, whilst the overall argument which combined these sub-arguments might itself—being relatively abstract by nature—simply utilise classical propositional logic with relatively simple and straightforward use of features.

4.2 Meta-level reasoning over dependability arguments

A language which does accommodate the above features, attributes, and logical and extra-logical operators, provides an object language in which dependability arguments become complex statements—theories, if one will. However, while such an object language allows us to construct and validate arguments, we also require the ability to reason *about* such arguments. We need a language in which we may express and reason about the properties and qualities of arguments, such as their validity or 'strength', and in which we may manipulate, transform and reuse arguments. A meta-language, a (partial) language in which the atomic objects are not evidence but are entire first-order arguments themselves, provides a framework in which we may explore such issues.

In a meta-framework, dependability arguments would be first-order terms in the language. Sentences in the language would permit us to reason about these arguments; in particular, to reason about,

- the *properties* of dependability arguments, such as their validity and simplicity, or the perceptions and assumptions that underly them.
- the *quality* of dependability arguments: are sub-arguments independent; how robust is an argument; does it critically reuse or depend upon particular evidence?
- *transformations* of arguments, such as simplifications or restrictions to particular viewpoints, and reasoning over *classes* of arguments.
- abstracting over common patterns of arguments, to support *reuse* and to understand under what constraints and contexts such reuse is valid.

Let us take viewpoints as an example to illustrate the need for, and benefits of, such a meta-framework. Consider the situation in which a systems developer

constructs and presents to an assessor the dependability case for some proposed system. Let us refer to the representation of the argument in this dependability case, in our argument representation object language, as A0. Let us assume that, as it stands, this argument is (logically) valid. However, suppose that the assessor, whilst inspecting the argument A0, determines that at certain points it uses an appeal that the assessors find unsatisfactory. It might be, for example, that some authority is cited in whom the assessors have insufficient confidence. In this case the assessors would wish to inspect whether the argument remained valid if the confidence in these appeals was significantly weakened—or even rejected entirely. The assessors take a particular *viewpoint* of the argument, in which certain evidence or warrants for conclusions are weakened or rejected. This viewpoint is, in effect, a transformation of the argument A0 to a new argument, A1, which may in fact not be valid. If it is the case that A1 is not sufficient to satisfy the authorities then it would be returned to the developers who may then transform the argument again, to a new argument A2, in which the weaknesses of A1 have been addressed—possibly by the addition of extra evidence or some other strengthening of the weaknesses highlighted by the transformation from A0 to A1. A meta-framework in which we may represent arguments as first order terms, permits us a means of expressing a variety of such transformations of these arguments, and of reasoning over the properties of these transformations—whether they preserve (logical) validity, for example. Furthermore, representing arguments as first order terms permits us to reason over partial instantiated patterns or fragments of arguments, and thus assists in the understanding of the viability and consequences of the reuse of arguments, such as that currently studied by Smith and Harrison (2002).

5. Conclusions and future work

The study of dependability argumentation is both a rich and interesting area. Dependability arguments are often complex in nature, combining disparate forms and sources of evidence in detailed arguments, which must then be reviewed by, and negotiated with, a broad audience exhibiting diverse competencies and interests. The issue of *representing* dependability arguments is a pressing one, and current approaches—while often expressive—generally lack sufficient semantics or depth to provide the necessary support for the analysis and evaluation of arguments.

Adopting a formal approach to representing dependability arguments has many advantages, most notably in the precision and exactness that such an approach brings to questions of the meaning and validity of an argument. However, given the relative sophistication and detail contained in a typical dependability argument, the expressiveness required of any language for representing such arguments goes significantly beyond what typical formal logics presently offer. Our review of safety cases has explored this topic, and illustrates what is required of a representation for it both to be sufficiently expressive, and to support the variety of evaluations and manipulations demanded of dependability arguments. Studies of the

representation of dependability arguments have much to learn from the study of Informal Logic, an illustrative example being the study in Informal Logic of issues of validity and plausibility with regard to appeals to authority—a major aspect of many safety cases.

We propose that, to be sufficiently expressive, a (formal) representation of dependability arguments requires both an object language and a meta-framework. At the object level, arguments may be constructed as theories in an extended logical language: where evidence captured as atomic propositions, with associated identifying labels and sets of features, and where complex statements are constructed by combining these propositions using both logical and extra-logical operators. At the meta-level, these representations of dependability arguments as extra-logical theories become themselves the objects of discourse: permitting reasoning over their various qualities such as validity, strength, robustness and the like; and permitting the manipulation and transformation of such arguments whilst maintaining strict control over the constraints under which these transformations are considered permissible. Such a meta-framework is an essential requirement for the foundations of the argumentation tools which are equally necessary to support dependability arguments.

The formalisation of such a representation of dependability arguments is the subject of ongoing study. At present the syntax of both the object and meta-languages is available and work is ongoing concerning the semantics, both model-theoretic and proof-theoretic, for these. As the saying goes: the proof of the pudding is in the eating, however. Consequently we must explore the sufficiency of the expressiveness of this representation through its application to practical examples of the modelling of dependability arguments, such as the EUR RVSM safety case illustrated in Figure 2. Such application will permit us to explore important questions: such as whether further features or attributes are required for claims; and whether further extra-logical warrants are necessary. The application of our representation in this manner will not only assist in the evaluation of its expressiveness but will also teach valuable lessons concerning the requirements of tool support for dependability argumentation, and the requirements for the transformation, manipulation and reuse of dependability arguments.

Consideration of tools to support dependability argumentation brings us to one final issue that we consider important for future study. We have considered at some length here the requirements for a sufficiently *expressive* representation of dependability arguments, yet we must also consider what such a representation requires to be sufficiently *effective* for the human user. This is to say, the object language and meta-framework we propose offers a sufficiently expressive form of representing dependability arguments, but a formal description of such an argument may be confusing or unconvincing unless it is presented in a form which is (or forms which are) accessible to the broad range of reviewers who must assess it. This issue, as it relates specifically to dependability arguments for safety-critical

systems, is discussed at length in (Gurr 1997). A conclusion of that discussion is that salient and accessible presentations of the high-level structure of dependability arguments, such as those which the diagrammatic ASCE and GSN notations reviewed in Section 2 seek to capture, are an essential part of an effective argument representation.

Diagrams and new diagrammatic languages are frequently cited as being "natural" and "intuitive" notations, permitting "easy" and "accurate" communication of complex structures and concepts. Indeed, such claims have been readily applied to the many graph-based notations popular in dependability argumentation, as reviewed in Section 2. However, the veracity of these claims is seldom tested and often the design of such diagrammatic languages follows no clear or obvious principles of usability, readability or effectiveness for the human user. Recent work in Gurr (1999) and Gurr and Tourlas (2000) addresses these issues, drawing together results from formal methods, visual language theory, cognitive science, empirical psychology and graphic design in order to deduce guidelines for the design and use of new diagrammatic languages, and for the construction of diagrams in existing diagrammatic languages.

Building upon this foundation we intend to supplement our languages for representing dependability arguments with effective diagrammatic languages, thus providing natural and intuitive representations of arguments for human readers. After all, it is clear that ultimate responsibility for acceptance or rejection of any dependability argument will always rest with a human agent or agency. As such, it is both in our interests and within our responsibility to ensure that these agents or agencies are presented with the most comprehensive, comprehensible and accessible representations of dependability arguments that it is within our power to provide.

Notes

[1] EUR-RVSM Pre-Implemtation Safety Case. Edition 2, 14 August 2001. Available from: http://www.eur-rvsm.com/library.htm.

[2] *Cf.* Endnote 1.

[3] ASCE (Adelard Safety Case Editor). Homepage: http://www.adelard.com/software/asce.

[4] IEC 61508. *Draft International Standard 1508 "Functional Safety: Safety Related Systems"*. Available from British Standards Institution, BSI Sales Office, 389 Chiswick High Road, London W4 4AL, 1995.

References

IEC 61508. *Draft International Standard 1508 "Functional Safety: Safety Related Systems"*. Available from British Standards Institution, BSI Sales Office, 389 Chiswick High Road, London W4 4AL, 1995.

ASCE (Adelard Safety Case Editor). Homepage: http://www.adelard.com/software/asce.

Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, (Eds.). (2002). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge: Cambridge University Press.

Carroll, J.M. and T.P. Moran (Eds.). (1991). Special issue on design rationale. *Human-Computer Interaction*, 3 & 4 (6).

Conklin, J. and M.L. Begeman. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6 (4): 303-331.

DIRC: Interdisciplinary Research Collaboration on the Dependability of Computer-Based Systems. Homepage: http://www.dirc.org.uk.

EUR-RVSM Pre-Implementation Safety Case. (2001). Edition 2, 14 August. Available from: http://www.eur-rvsm.com/library.htm.

Fox, J. and S. Das. (2000). *Safe and sound: Artificial Intelligence in Hazardous Applications*. Cambridge, MA: MIT Press.

Gabbay, D.M. (1966).*Labelled Deductive Systems: Volume 1*, volume Oxford Logic Guides 33. Oxford: Oxford University Press.

Girard, Jean-Yves. (1995). Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, (Eds.), *Advances in Linear Logic*, pp. I-42. Cambridge: Cambridge University Press.

Gurr, C. (1997). Knowledge engineering in the communication of information for safety critical systems. *The Knowledge Engineering Review*, 12 (3): 249-270.

Gurr, C. and K. Tourlas. (2000). Towards the principled design of software engineering diagrams. In *Proceedings ICSE 2000: 22nd International Conference on Software Engineering*, pp. 509-518. ACM Press.

Gurr, C.A. (1999). Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *Journal of Visual Languages and Computing*, 10 (4): 317–342, August.

Lee, J. (1993). Design rationale capture and use. *Artificial Intelligence*, 14: 24-26.

Lee, J. and K. Lai. (1991). What's in design rationale? *Human-Computer Interaction*, 6 (3 & 4): 251-280.

MacKenzie, D. (to appear) A Worm in the Bud? Computers, Systems, and the Safety-Case Problem. In *Proceedings of the Dibner Institute Symposium on "The Spread of the Systems Approach"* (to appear), Cambridge, MA, May 1996.

Moran, T.P. and J.M. Carroll (Eds.). (1996). *Design Rationale: Concepts, Techniques and Use*. Hillsdale, NJ: Lawrence Erlbaum Associates.

University of York, CAA, and Logica Cambridge Ltd. (1991). SAM—A Tool For Reasoning about Safety. Objectives, Requirements and Operational Concept.

Rittel, H. and M. Webber. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4: 155-169.

Buckingham Shum, S. (1996). Design argumentation as design rationale. In A. Kent and J.G. Williams, (Eds.), *The Encyclopedia of Computer Science and Technology*, volume 35, Issue 20, pp. 95-128. New York: Marcel Dekker Inc.

Smith, S.P. and M.D. Harrison. (2002). Improving hazard classification through the reuse of descriptive arguments. In C. Gacek, (Ed.), *Software Reuse: Methods, Techniques, and Tools*, number LNCS 2319 in Lecture Notes in Computer Science, pp. 255-268. Berlin: Springer.

Walton, D. (1984) *Logical Dialogue-Games and Fallacies*. Lanham, MD: University Press of America.

—. Walton. (1989). *Informal Logic*. New York: Cambridge University Press.

Wilson, S.P., T.P. Kelly, and J.A. McDermid. (1995). Safety case development: Current practice, future prospects. In *Proceedings 1st ENCRESS/12th Annual CSR Workshop*. Springer-Verlag.

C Gurr
School of Informatics
The University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK
Email: C.Gurr@ed.ac.uk