

4 Making the Case that its Safe

Before a safety-critical system, computer-based or otherwise, is approved for service, the developers are now commonly required to present a convincing argument to the appropriate regulatory authority that the system will operate in an acceptably safe manner. Such an argument has come to be known as a *safety case*.

Although there is variation of presentation and content of safety cases in different industries and regulatory authorities, a good working definition of a safety case is the following: “the purpose of a safety case is to present a clear, comprehensive and defensible argument, supported by calculation and procedure, that a system or installation will be acceptably safe throughout its life and decommissioning”. A safety case should demonstrate that all the hazards generated by the environment in which the software will operate and the hazards created by the software itself failing to operate as required have been identified and assessed and that any necessary modifications have been made. It is a document which evolves at each stage in development as knowledge of the software increases. All the lifecycle products which contribute to the safety of the software need to be brought together into a software safety case.

Safety cases tend to be very large documents, containing complex internal inter-dependencies, which include the results of a wide range of related analyses. They often rest upon a number of implicit assumptions and have a long lifetime, going through many versions in the course of their production. Both product and process issues need to be addressed in the safety case; it must be shown both that the system meets its safety requirements and that the processes for deriving the requirements, constructing the system and assessing the system are of appropriate integrity. The safety analyses which appear in the safety case depend crucially upon the formulation of suitable models of the system, at various levels of abstraction, produced during the development process. Given these characteristics, it is not surprising that safety cases are difficult and expensive to produce and maintain.

A major problem with many current safety cases is that they contain a great deal of evidence from the various safety analysis techniques employed, but they do not always draw this evidence together in a clear and understandable manner. The ASAM II project, which set out to provide a structured method and comprehensive tool support for the production of safety cases, therefore decided that it was necessary to present both high level arguments as to why a system will be acceptably safe in operation, and the detailed supporting evidence to back up the high level arguments.

4.1 A method for safety cases

The approach adopted by the project was to use the idea of a *goal hierarchy* to present the high level arguments. The goal hierarchy thus forms the “spine” of the safety case, with the supporting evidence, argument and calculations being attached to this spine. The goal hierarchy is supported

by design models using appropriate notations, and the various hazard and safety analysis techniques are in turn expressed in terms of these models.

In current safety cases, there is rarely complete consistency between the different safety analyses used, and between the design models as used in the safety case and the design as used for constructing the system. These inconsistencies tend to make the safety case less than wholly defensible. To remove these inconsistencies it is necessary to decide what the consistency rules should be. This is quite a complex task, because several of the techniques fulfil more than one role in the safety analysis process, and there is usually more than one legitimate way of using any given technique. Thus detailed study and development of a theory underpinning the analysis was required. This work involved analysing the relationships between a number of commonly used safety analysis techniques, for example Fault Tree Analysis (FTA), Failure Modes and Effects Analysis (FMEA) and HAZOP (Hazard and Operability Studies). From this analysis, the project has developed the following:

- a set of data propagation rules to identify how various techniques are related, for example an effect in an FMEA becomes a leaf event in a FTA;
- a set of consistency rules between ten major safety analysis techniques used by the industrial partners, and supported by the SAM tool;
- a unifying data model which enables the effective and efficient expression and implementation of mechanisms for checking conformance to the consistency rules, and analysing the effect of changes to system designs or analyses.

Tool support

The Safety Argument Manager tool, SAM, supports the construction and editing of goal structures, a number of system modelling techniques (a general modelling notation, using simple block and line diagrams, data flow diagrams and state transition diagrams and tables), and a number of analysis techniques (HAZOPS, FMECA, FTA, Functional Failure Analysis and Event Tree Analysis). A hazard log provides an index into the safety case. Links may be set up between a goal hierarchy and models or analyses documents, so that navigation to the appropriate part of the safety case data is possible. System models constructed using the model editor can be referred to by various analysis techniques, and partial automatic generation of safety analyses from a model is possible. At present, models built from data flow and state transition diagrams are for information only, and are not connected to the safety analysis technique, but this will be a topic for improvement in future versions of the tool.

Graphical, tabular and textual notations are provided, and some correctness rules are enforced or checked. In the fault tree editor, for example, it is not possible to make gate-to-gate connections.

The SAM tool allows any part or parts of the safety case data to be selected as “foils” and linked into Microsoft Word documents using Object Linking and Embedding (OLE2) such that when the safety case data is changed via SAM the corresponding changes are reflected in the linked documents (the PC platform was selected for the tool on the advice of the industrial partners, on the grounds that this platform would have the widest acceptance in the industries that have to produce safety cases).

The safety case data is held as a series of “documents” stored as individual files; each document holds a particular notation, such as a goal structure diagram or a HAZOP table. An underlying database, implementing the method data model, is used to summarise the safety case data

structures and to allow analyses to be checked for consistency. The SAM tool also uses this database to provide partial automation of the generation of safety analyses, including regeneration following changes. In this regard SAM is comparable to other more conventional CASE tools, but seems to be unique in its support for integrated safety analysis techniques.

Case studies

A number of case studies have been carried out, using the SAM tool, both to validate the underlying method and to provide feedback on the facilities of the tool and any problems encountered in using it. The case studies were carried out by the industrial partners, with method and tool support from the University of York, with the exception of one study which was undertaken jointly by the University and BAe Military Aircraft.

The case studies considered an aircraft landing gear control system, a gas turbine engine monitoring system, a deep mine ammonia refrigeration plant, and a liquid-conditioning system for a jet pilot's flying suit. Between them, they have exercised most of the method concepts and tool facilities, including all of the safety analysis techniques.

The case studies have been very valuable both in verifying the overall effectiveness of the method supported by SAM, and in identifying necessary improvements to the tool.

4.2 A Designer's Assistant

A good part of the evidence which needs to be collected to support a safety case is generated during the process of design. Unfortunately, little software support exists for gathering and compiling this evidence. Retrospective assembly of the evidence requires skill and insight when reviewing the work, and is very time-consuming. The SCIDS project has therefore constructed a *Designer's Assistant*, which addresses these problems by gathering information on the designer's work and linking this information to form a framework of *design concerns*. The *Designer's Assistant* works in conjunction with the *Online Design Journal* (ODJ) to provide integrated support for all 'paper based' design methods since the ODJ is an analogue of the designer's daybook. This provides basic sketcher and notepad facilities modelled closely on the behaviour of the paper-based day book. The range of the support extends into the CAD/CASE arena as any Windows based design tool can be used in the computing environment provided by the *Designer's Assistant*.

In general, design concerns arise at each stage of the project design life-cycle and originate in the quality and safety goals allocated to products and processes. At the outset, a top-level set of concerns are either stated in the user requirement, arise from a hazard analysis, or represent

Documentation and software

Further details of the SAM method and tool can be found in the document *ASAM II: Concepts and Process* available from York Software Engineering [ASAM II ref 1]. The Final Report of the project is also available; this describes the project and its results and provides a list of published papers.

The SAM tool and its User Guide are available on licence or other suitable terms from YSE.

YSE will seek to continue the development of the tool in conjunction with the University of York and to use it in their consultancy work. The industrial partners have plans to introduce the SAM concepts, and the tool itself, into their own work. A SAM User Group will be set up to share experiences with using the tool and to steer its further development.

enterprise-related constraints and goals. As design proceeds and concerns are addressed, the design solutions cause designers to identify lower level concerns which taken together, in subsets, or singly, satisfy higher concerns. Other concerns arise at each stage to do with features of the engineering solutions adopted. To argue that a design is safe one must supply evidence showing how each of the concerns has been addressed and the way they are related to produce a safe system in the form of a reasoned safety argument, the safety case.

The Designer's Assistant is a *persistent user agent* that has access to enough detail of on-going design activity to characterise the work and to identify the concerns being addressed. It may also solicit additional input from the designer. This access to design detail is seen as the basis for a range of services: The following are currently being implemented:

- *Automatic record of work.* A detailed record of work is assembled to supplement the records kept by designers, to support the designer in supplying the information required by management in project control, planning or for audits.
- *Classification of work by concern.* Where the design concerns are predefined, as required by many safety critical systems guidelines, tasks can be classified by the concerns they address. The Designer's Assistant lists the concerns that have been addressed and highlights those yet to be addressed, indicating which tasks need to be carried out, providing both a valuable *aide memoire* for the designer and a constantly updated, concise, indication of project status as regards the safety argument in particular.
- *Memory prosthesis.* When the Designer's Assistant detects that a concern is being addressed, documents containing information relevant to the current work are searched for and links provided to them. Such links may thus be to
 - earlier design work,
 - relevant on-line standards,
 - guidelines, or
 - help files on the problem domain.

An important feature of the Designer's Assistant is that it is *not prescriptive*. Most computer support for designers is in the form of CASE or CAE tools, typically requiring the use of specified methods. Here the designer is free to use any tools and methodologies approved by the organisation.

The role of the designer

The designer's role is one of *originator*, undertaking work in repines to a *register of concerns*. The role of the Designer's Assistant, in contrast, is to marshal the evidence for a safety case and other supporting documentation. The designer remains responsible for updating the register of concerns and monitoring the accuracy of the Designer's Assistant's work..

The Designer's Assistant would have to capture, identify and characterise all the designer's actions to provide fully comprehensive support. This is not possible because computer support for many aspects of designers' work is poor or non-existent and the Designer's Assistant cannot compile evidence on work that it cannot observe. Further, limitations of the hardware platform and the system software, mean that complete recording is not possible. So any evidence gathered by the Designer's Assistant must be considered as a draft based on what can be observed of those elements of design supported by the computer system. It must be the designer's responsibility to complete any Designer's Assistant report by adding details of that work not reported.

Register of Concerns

Concerns are central since the Designer's Assistant seeks to relate each item of design work with a concern by computing the degree of agreement between features of the work item and features of known concerns maintained in a *Register of Concerns* which the Designer's Assistant accesses for this purpose.

The characterisation of a concern by a set of features allows the Designer's Assistant to identify an episode of work as being associated with one or more concerns. At this stage, only keywords are being considered as features, although future work may extend this to recognition of the gross features of sketches. Thus each concern is labelled or characterised by a set of keywords. These keywords may be entered directly by the designer or by a safety engineer. Identification and definition of the concerns arises from the consideration of quality and safety issues relating to the product, its development and target environment This task could be carried out using the *Online Design Journal* or using, for example, one of a number of *Mind Map* tools. Each concern is named, associated with its distinguishing features, and placed in the register as it is generated. It is foreseen that, eventually, the register could be initialised by loading it with the information about concerns generated in earlier projects of similar scope.

In a multi-user version of the Designer's Assistant, the *Register of Concerns* would be project-wide, shared between all the designers working on a project, and, consequently, a common view of the status of concerns being addressed would be obtained.

Architecture

The Designer's Assistant runs under Windows 3.1. using a Novell 4.1 network and Lotus Notes v3.1. This provides for future application in a team context where project data is centralised and shared. The system is constructed as an agent, composed of a number of active and passive sub-components, as shown in the diagram. Since the ODJ has the status of a design tool that is being observed, it is not shown in this diagram.

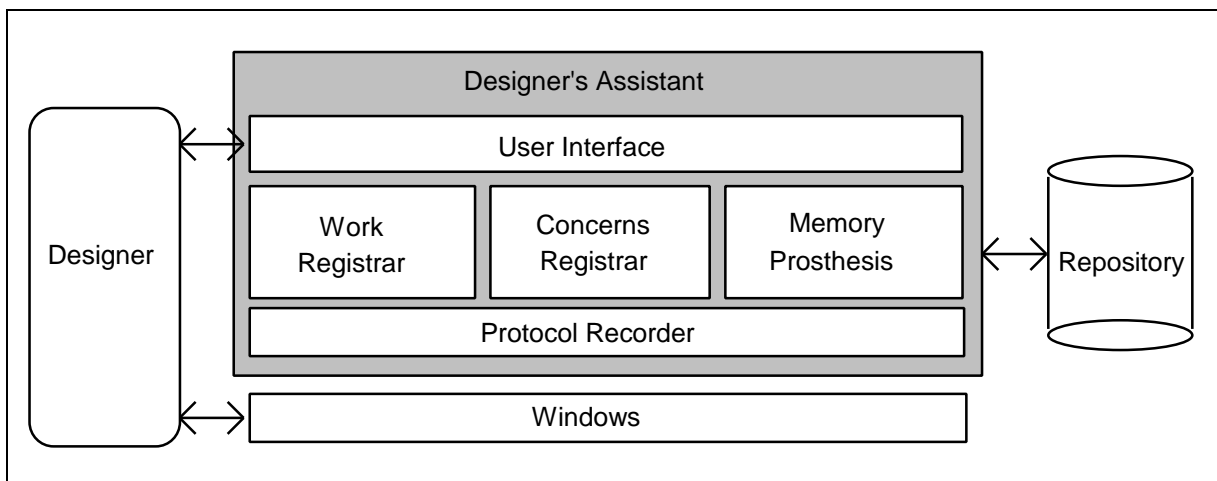


Figure 4.1. The Designer's Assistant

The *User Interface* is provided by the *Online Design Journal* when the designer is engaged in early sketch or narrative based design work. However, any Windows-based tool the designer

wishes to use may be invoked and used. The User Interface is designed to be unobtrusive so as to limit disturbance of the designer's work and consists of a small *status bar* and a dialogue window launched when appropriate. These provide the facilities by which the agent can report and negotiate its key decisions with the designer, appealing for clarification on the concern currently being addressed. Otherwise its interpretation appears on the status bar. The user may request that the agent correct its reported interpretation of the current concern by launching the dialogue window. Any corrective action is reflected in the status bar. The status bar also shows when the Designer's Assistant's memory prosthesis has identified documents which may be of use to the designer. The designer can obtain more information on, and view, the documents via a hyperbook-like mechanism, by point and click.

The *Protocol Recorder* captures the flow of information across the User Interface, by intercepting the Windows message stream to obtain keyboard messages as they occur. Thus, it responds to keys as they are pressed, attempting to derive keywords from character sequences (much as a compiler does) and hence derive the raw material by which concerns are identified. It also responds to the 'On idle' event, to achieve the identification of the current subject of the work by file and application name.

Each episode of design work carried out on a particular file using a particular application is recorded as a *work item record*, containing information such as: *user id*, *project name*, *date*, *time*, *work reference (tool reference, file reference)*, *work features (keywords)*, and *duration of work*. The *Work Registrar* assembles and archives these work items in the Record of Work. The *Concerns Registrar* interacts with the user to populate the Register of Concerns.

The *Memory Prosthesis* compares the current concern against the contents of the repository to fetch relevant files and data which the designer may find useful and presented by maintaining a hyperbook comprising a number of Virtual Journals. In relating concerns to episodes of work, the *Memory Prosthesis* considers the episode's features, principally paying attention to their application name, file name, and the keywords. To perform a match, the features of the episode are compared to those of the concerns in the register and the best match results in a candidate concern.

The Designer's Assistant in use

On completion, the Designer's Assistant will be evaluated by the research team and by the industrial partners.

The team believe that the information accumulated by the Designer's Assistant will provide significant assistance in assembling safety cases and audit evidence, and also in gauging the effectiveness of the quality system. The system will only be effective, however, if it is used continuously throughout a development project. As an extra incentive for this, the Designer's Assistant's memory prosthesis provides immediate help in reusing designs, and for transferring design knowledge to other designers.

Further background on the Designer's Assistant can be found in [SCIDS ref 1] and [SCIDS ref 2].

4.3 *Combining diverse evidence*

It must be remembered that a safety case is an *argument* - an argument as to why a particular panoply of technical and managerial approaches will identify all significant risks and will sufficiently mitigate the risks. This argument will usually be based upon a wide and diverse range of sources of evidence, including evidence that certain good management practices have been followed; evidence from quantitative, logical, or inspection analyses; evidence from failure data derived from observation of a system under test or in operation. Much of the evidence will be dependent on expert judgment - and there is some evidence of experts being unduly optimistic about their judgemental abilities.

Tools such as SAM and the Designer's Assistant provide extremely useful support for presentation and management of the mass of material required for a safety case. However, aggregating disparate types of evidence into an argument is akin to aggregating chalk and cheese. Again, the combination of evidence, and the credence and confidence which can be placed in the steps of the argument often entail yet more expert judgement. Even with tool support it is difficult for an outsider to assess the validity of such an argument.

The DATUM project was based on the belief that it might be possible to improve both the basis and the presentation of safety arguments by taking account of and combining, in a more rigorous way than has been done in the past, the many disparate types of evidence that might be available. This could include not only engineering judgement and failure data, but also evidence of the efficacy of the development methods used, experience in building similar systems in the past, competence of the development team, architectural details of the design (including especially the human computer interface), etc. The challenge for DATUM was to provide a rigorous measurement-based approach that could overcome the serious problems involved in combining disparate (and often uncertain) evidence in order to make a single evaluation of the overall dependability. A secondary objective for DATUM was to help developers determine how different development methods and system architectures contribute to the overall dependability argument.

Assessing dependability by combining evidence

A number of formalisms could, in principle, be used for modelling uncertainty. The project examined in depth the various methods and technologies, including Bayesian probability, Dempster-Shafer theory, fuzzy sets and possibility theory. [DATUM *ref* 3] provides a comprehensive overview and in-depth comparison of these approaches. The project identified the advantages and disadvantages of each method. For example, Dempster-Shafer's theory of belief functions was attractive since it can apparently deal with problems where the evidence is too weak to support a parametric model. However, this approach provides no capability to account for 'dependencies' between the two or more bodies of evidence which are combined, and there is a serious danger in the safety context of over optimistic assessment resulting from failure to recognise stochastic dependencies.

Although no single formalism for uncertainty was perfect for the project's purposes the result of their study was a decision to adopt the most mature and well-developed (and arguably the most convincing) namely *Bayesian probability*. [DATUM *ref* 4 and DATUM *ref* 5] provide specific justification for this in the context of systems dependability assessment. Bayesian probability theory has been extended beyond merely a means of representing uncertainty, and in fact

provides a coherent theory of how to *act* on the world in an optimal fashion under circumstances of uncertainty. Fuzzy sets and Dempster-Shafer theory do not as yet appear to offer anything comparable. Bayesian probability offers a language and calculus for reasoning about the beliefs that can be reasonably held, in the presence of uncertainty, about future events, on the basis of available evidence. *Prior* probabilities are thus updated, after new events are observed, to produce *posterior* probabilities. By repeating this process, the implications of multiple sources of evidence can be calculated in a consistent way.

Having chosen Bayesian probability the next task was to ‘put it into practice’. This providing a practical and usable method for assessors to use the formalism in order to take into account explicitly all the factors that affect the reliability of a software product: proficiency of developers, effectiveness of tools, effectiveness of inspections and debug testing, effects of specification and programming languages, specific difficulties of an application or a specific project, etc. The usual impediment to using this multiple evidence in Bayesian reasoning has been the overly complex computations that result. This has, in the past, severely restricted the scale of problems and evidence that could be tackled. However, a relatively new but rapidly emerging technology provides an elegant solution (complete with appropriate computer tools) enabling the project to push back the boundary of the problems that can be attacked: this is the technology of *Bayesian Belief Networks* (BBNs)¹.

Bayesian Belief Networks

A BBN is a graphical network that represents probabilistic relationships among events. BBNs enable reasoning under uncertainty and combine the advantages of an intuitive visual representation with a sound mathematical basis in Bayesian probability. With BBNs, it is possible to articulate expert beliefs about the dependencies between different variables and to propagate consistently the impact of evidence on the probabilities of uncertain outcomes, such as ‘future system reliability’. Two examples of BBNs are shown in figure 4.2.

¹ Referred to in section 11.1 as *Causal Nets* or *GPMs*

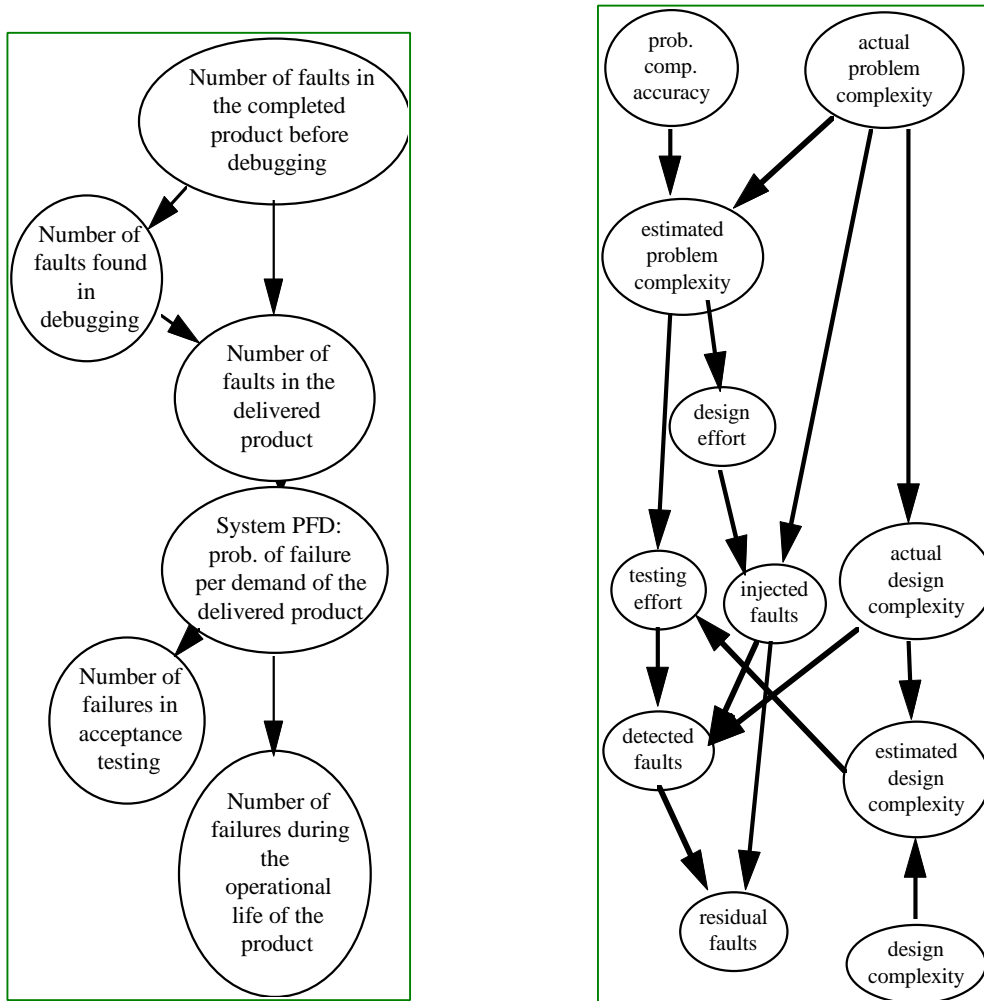


Figure 4.2. Example BBNs used in DATUM

The BBN on the left uses comparatively little evidence, depending only on the observed reliabilities and defect counts of previous products of the same process, and on the defects discovered in the current product during debugging. The topology of the graph is used to indicate probabilistic relationships among the variables described in the nodes. BBNs allow an injection of scientific rigour when the probability distributions associated with individual nodes are simply “expert opinions”. A BBN will derive all the implications of the beliefs that are input to it, and some of these implications are statements of fact that can be checked against the observed reality of a software project, or simply against the experience of the experts and decision makers themselves. The BBN on the right includes subjective indicators, like problem complexity and design effort. Thus, this network is meant to be populated with probabilities that are not all derived from statistical inference, but at least in part from expert opinion. The advantage of using BBNs is then that of checking the consistency of individuals’ beliefs and decisions.

The use of BBNs makes the decision process easier to describe, and thus to check, communicate and audit. This was borne out in the results of a case study in which assessors’ arguments about a system’s dependability and safety were described in terms of specific BBNs. The results were that assessors were able to better characterise and communicate their complex webs of inference, and the very act of producing a BBN in conjunction with an expert in the technology resulted in refinements and improvements to the safety case.

Because BBNs have a rigorous, mathematical meaning there are software tools that can interpret them and perform the complex calculations needed in their use. The specific tool used in DATUM was *Hugin Explorer* which provides a graphical front end for inputting the BBNs in addition to a computational engine for the Bayesian analysis. The resulting framework for dependability assessment is described in [DATUM ref 6, 7, and 9]).

Obtaining the probabilities

BBNs allow rigorous reasoning with uncertain knowledge, irrespective of whether abundant empirical data is available. However, while BBNs, assisted by software tools, allow us to make the best use of the complex evidence available in predicting software dependability, the accuracy of the predictions is ultimately improved if certain empirical data is also available. For example, the BBNs in the figure above require prior probabilities of numbers of faults or fault densities at different development stages. While purely subjective probabilities could be used in the absence of real data, the latter is clearly preferable.

To address this, DATUM provided two types of guidelines for ‘populating’ the probability tables of BBNs. On the one hand [DATUM ref 9] describes how relatively simple measurement programmes could provide the necessary data directly. A well managed software development project has a wealth of potentially important quantitative information to support safety assessment. This could be: failure data from testing; fault and change reports from various project phases; test results including coverage measures; outputs from static analysis or metrics tools; various internal system quality indicators such as modularity measures; measures of effort associated with various project phases, and, last but not least, historical evidence of efficacy of tools and techniques. In particular good data about software defects and changes, with adequate records of previous projects, can provide important quantitative information on which to base a reliability assessment or safety case.

On the other hand the project also produced a range of results that enable assessors to use quantitative data even if they have none available from their own projects. For example, they provided:

- a wealth of data on fault densities after examining all publicly available data (see [DATUM ref 10]);
- data gathered from public sources on the likely impact on reliability from using formal methods during development ([DATUM ref 11]);
- evidence about the number of representative test cases required to obtain a high confidence (99%) that the probability of failure on demand is smaller than various specified limits ([DATUM ref 12]).

Multi-Criteria Decision Aid

For all their many benefits, BBNs do require problems to be fully described in probabilistic terms. This may be difficult for an untrained user. Thus DATUM investigated (and also applied) methods whereby probability distributions could be extracted from untrained users by extrapolating from a small number of assumptions [DATUM ref 6].

Nevertheless, there are a range of alternative rigorous methods for structuring complex decisions for situations in which the available knowledge is intrinsically difficult to express in probabilistic

terms. Most notable among these is the rapidly evolving field of Multi-Criteria Decision Aid (MCDA). DATUM considered the extent to which MCDA could be usefully applied to the specific problems of dependability assessment [DATUM ref 8].

It turns out that there are three classes of methods which come under the umbrella of MCDA. The most well known is *multiple attribute utility theory* (MAUT). All the methods in this class attempt to optimise some utility function, defining a strict order over the set of all possible decisions. This approach has been mainly used within the business quality community but it has also been used in assessing dependability of PLCs and to assess (theoretically) the ‘best’ way to improve safety on the space shuttle. The problem with MAUT is that, in forcing a strict order over the set of decisions, it makes very strong assumptions about the underlying criteria; these assumptions are generally unrealistic for software dependability assessment.

The project therefore experimented with a second class of MCDA methods called *outranking methods*, which depend on much less stringent assumptions. The result is that you get a partial (as opposed to a strict) order over the set of decisions. This then means that your choice is narrowed down to the set of decisions which are optimal in the partial ordering. The final class of MCDA methods are the interactive methods whereby the set of decisions is incrementally narrowed by interactive techniques (after each ‘round’ the decision maker is asked to input additional information). These methods allow the decision-maker to articulate requirements and decision criteria, and choose among available alternatives in a sensible way: while they do not offer “optimality” in the same sense that Bayesian decision theory does, they do prevent most of the inconsistencies commonly associated with such decision-making. MCDA techniques appear to deserve further investigation, especially in the way they could be integrated with BBNs.

Further development

The DATUM approach to dependability assessment using BBNs is being pursued by several major European organisations involved in developing and/or evaluating safety-critical systems, under the ESPRIT project SERENE. This project is producing a BBN-tool based on Hugin that is specifically tailored for safety assessment. In the UK the results of DATUM have already impacted the new MOD defence standard 00-40 on reliability and maintainability of systems involving software. Moreover, the Defence Research Agency has recently begun funding a collaborative project with City University to use BBN technology to improve its predictions of reliability of complex vehicles and equipment.

In addition to the more immediate exploitation of the DATUM technology City University has also received funding to extend its basic research in this area. Specifically we cite:
Work is also continuing on further investigation of the important links between BBNs and MCDA and, on extending the use of BBNs for system quality evaluation in the commercial software domain.

Evidence from ‘similar’ products

A very common scenario is where increased confidence in reliability is claimed as a result of evidence from previous experience of ‘similar’ products (in addition to evidence from testing the product itself). [DATUM ref 12] examines this in depth. Importantly (but rather depressingly) they showed that this kind of additional evidence can only improve confidence in the reliability of a product quite modestly. They used a Bayesian model in their analysis. Essentially, if you

were to claim that great trust could be placed in a particular system because of past experience of other systems, you would have to justify this by trying to claim that your prior distribution is reasonable within the model.

The argument from diversity

Improvements in reliability are often expected as a result of using diversity in the product design. Recent models for the failure behaviour of systems involving redundancy and diversity have shown that common mode failures can be accounted for in terms of the variability of the failure probability of components over operational environments. Whenever such variability is present, we can expect that the overall system reliability will be *less* than we could have expected if the components could have been assumed to fail independently. In [DATUM *ref 2*] a well known model of hardware redundancy is generalised and it is shown that, with forced diversity, this unwelcome result no longer applies: in fact it becomes theoretically possible to do *better* than would be the case under independence of failures (although in practice this is likely to be very rare).

Diversity arguments were also explored in the Network Programming project (see section 15.3).