5

Requirements, Claims and Evidence

5.1 Where to Start? The First Foundation of Dependability Justification

"The only way to get somewhere, you know, is to figure out where you are going before you go there" says the farmer to Rabbit in John Updike's "*Rabbit, Run*" [101]. This common sense advice takes on a special meaning in the context of dependability, because the point of departure of the justification is also its destination. We regard a *dependability justification* as the set of arguments and evidence components which support a carefully selected set of requirements addressing the dependability of the operation of a computer-based system in a given environment, for instance an industrial plant. These requirements need careful attention; they are the raw material we start off with and also the properties we have to justify in the end.

For reasons that will become clearer later, these requirements will be referred to in this book as being the *initial dependability requirements or initial safety, availability, security… requirements:*

Definition 5.1 *Initial Dependability Requirements* are specific to the application supported by the computer-based system and to the environment of the system. However, and this is important, they do not assume the use of a computer system. They specify *what* has to be achieved in the environment under specific constraints, not *how* it must be achieved. Basically, they require that some properties be maintained in the environment (*e.g.* an industrial plant) under specified constraints, whatever system is to be implemented.

The *specification* of the *initial dependability requirements* is the role of application, plant and/or safety experts; these experts, and only they can and must establish and validate the properties and the relations that must be maintained in the environment. No computer expert need be involved at this stage. Contrary to certain current practices, one could perhaps even argue that they should preferably

take no part in this specification so as to keep a neat separation between the specification of what an ideal system is expected to do and the constraints inherent to a real implementation. This separation of concerns guarantees that the ideal system specification remains a non-ambiguous reference against which to validate the actual system.

5.2 The Initial Dependability Requirements (CLM0)

Two types of *initial dependability requirements* can be distinguished. *Functional requirements* specify – among other functions – the detection and control of hazards, the necessary protection and prevention functions, safety and security actions, inventories, alarm generation, *etc.*

Non-functional requirements specify the properties that any implementation of these functions should satisfy; accuracy, stability, testability, maintainability are typical attributes that may be required. Targets on properties of the system behaviour within its environment and during operation such as timeliness, reliability, and availability are also possible requirements.

Initial Dependability requirements may therefore be expressed as deterministic or probabilistic assertions. But, contrary to what often is unfortunately the case, the *dependability requirements* must not be confused with the qualities required – *e.g.* by the state of the art or by standards – from the design or the project development processes. Such expected qualities are not of lesser importance but they are not part of *what* has to be justified; they are part of the justification. They are a *means* to provide evidence. They should be stated in separate quality assurance and V&V plans. The quality of the development and the V&V processes achieved by observing these plans contribute to producing or improving the evidence needed to support the claims into which the dependability requirements have to be expanded at the various levels of the implementation.

Initial dependability requirements may be formulated in a natural, informal or application-oriented language. In order to make things concrete, let us give two examples drawn from the case study in appendix B:

no_displacmt.clm0: Horizontal vehicle and vertical lift displacements are prevented when pallet is being inserted or extracted from a cabinet (*cf.* requirement (B1), page 297)

pfd_clm0: The probability of failure per demand shall be less than 10^{-4} per handling operation

In the first example, a specific functionality is required from the system, namely a protection function ensuring that the displacements are functionally impossible while handling a pallet of nuclear material. The second requirement is *non-functional*. It requires that the system enjoy a certain property, in this case the achievement of a reliability level. The first requirement is of a deterministic nature; the second one is probabilistic. None of these requirements refers to a computer. A more detailed elaboration of initial dependability requirements will

be given by the embedded system analysis of Chapter 9 (specifically in Section 9.3).

Initial dependability requirements are the foundation of the justification and considered as constituting its level-0. Their documented specifications shall be conveniently referred to as being level-0 statements, collectively denoted **CLM0.** Although these initial requirement specifications are data that are given as input to the justification, and not claims, this notation is chosen for reasons of simplicity and homogeneity that will become clear later. Each requirement should be identified by a mnemonic name and by its level, its statement being labeled in this book by a symbolic short-hand label <**name.clm0**>.

5.2.1 PIE's, Constraints, Safe States

In addition, the **CLM0** specifications must be based on and include a precise description of the environment; precisely:

- All *events* that can be generated by the environment and can affect the system behaviour;
- The *constraints* and restrictions imposed by the environment on the system behaviour;
- The *safe* environment *states.*

These *events*, often called *postulated initiating events*, must be described with their consequences and effects. By definition, these events and only those are the events that will have to be controlled by the system or that are assumed to affect its behaviour. The monitoring, control and mitigation of these events are primary – and often the only – functions of a protection or safety system. They correspond to hazards, causes of damage and failures that may occur in the environment; they may be generated by the failure of a motor, by a pipe break, the malfunctioning of a valve, or a loss of power. They may also be natural events such as earthquakes, floods, tornadoes causing electromagnetic or seismic interferences; also human–induced events such as plane crashes, ship collisions and terrorist attacks.

Associated with the events, of utmost importance is the **CLM0** specification of the states of the environment that are safe or unsafe. The system should indeed be able to deal with the *postulated initiating events.* The **CLM0** requirement specifications should therefore include the precise description of the *safe or preferred states* into which the system should be driven.

Secondly, the environment – and especially previously installed systems – imposes *constraints*, restrictions and invariant relations that must be maintained in the environment. Examples of such constraints are mass, energy or flow conservation laws; constraints imposed by the behaviour of pre-existing equipment. For example, a typical constraint of nuclear material handling systems (Appendix B) is the need to respect conditions on mass storage and locations determined by considerations on mass nuclear criticality or by limitations on radiation release.

We will see in Chapter 9 that models play an essential and concrete role in the description, documentation and validation of **CLM0** specifications. Particular details will be found in Section 9.3.

5.2.2 Elicitation

"*Principiis obsta*" (Beware beginnings!) wrote Ovid (43BC–17AD); and the artist Piet Mondrian said once that the most difficult brushstroke in a painting is the first one because it is unconstrained. The initial stages of a dependability justification are no exception. The definition and formulation of a complete and coherent set of *initial dependability requirements* has to be carefully done; their specifications need to have received the prior approval of all parties involved in what we referred to as the *dependability case*, possibly including the green light from safety and legal authorities.

This definition and formulation process always takes place in a given *pre-existing* context and environment. The process must take into account the assumptions and constraints dictated by the environment. Models and assumptions therefore play an important role which will be discussed at length in Chapter 8.

The **CLM0** specifications, independent of the actual implementation and those alone, determine and define what the safety of the computer based system is expected to be. Their formulation and truth are taken for granted in this book; so are the assumptions and the models on which these initial specifications are based. Since they originate from prior safety plant and environment analysis work, *validating* this set of initial specifications, *i.e.* asserting the completeness, soundness and coherency of the set, is outside the scope of this work. Plant engineers, regulator and Technical Support Organizations (TSO) specialists must do this critical task in the context of a safety analysis which is a primal and preliminary step of the computer dependability justification.

Of course, these initial requirements do not come from nowhere. The material from which they are elicitated and formulated consists of prior safety analyses reports, operational reports, accident reports, regulations, documentation of the system to be replaced or modernized, initial descriptions of the functionality, of the dependability requirements, and of the replacement constraints. A typical sample of this initial documentation is given for the SIP system in Appendix A.

In addition, these requirements are dictated by well-established *fundamental safety concerns or safety goals* specific of the application domain. For example, the design of the protection system of a nuclear reactor has to address the three fundamental concerns:

- Safe shutdown of the reactor and safe maintenance in the safe shutdown state;
- Removal of the residual heat;
- Prescribed upper limits for the exposure to radiation of personnel and environment.

And the first two concerns, for instance, are those that determine dependability requirements on the properties of specific subsystems of the protection system:

- The reactivity control subsystem;
- The reactor coolant inventory subsystem;
- The primary to secondary side heat transfer subsystem;
- The primary side pressurizing subsystem;

- The primary heat removal subsystem;
- The secondary heat removal subsystem.

5.2.3 Completeness

Hence, advice on how to elicit the initial requirements is basically outside the scope of this book. Computer experts need to understand these requirements but are not competent to define them. This important point is often misunderstood and source of problems. Some computer engineers believe they have to take the responsibility of defining the dependability requirements, confusing them with computer system requirements. Others tend to ignore them in the belief that dependability is nothing else but a computer system property. These two extreme attitudes are not rare but wrong and dangerous.

However, from the viewpoint of a "user", some general criteria can help the elicitation of dependability requirements. To maximize efficiency and pertinence, and to minimize cost, initial requirements must be, as much as possible, limited to functional and non-functional specifications that:

- Address the expected system functionality and dependability,
- Are dictated by the constraints of the environment,
- Are dictated by the use of specific technologies imposed by the environment.

As we shall see in Section 7.6, the attitude coined "enlightened catastrophism" may also provide useful viewpoints to identify initial dependability requirements. In any case, these requirements should be limited to those that are justifiable. This is a delicate issue which will be discussed and will become clearer later.

It is good practice that once defined and approved, initial requirements should be made explicit at the start of the project, and communicated to stakeholders, preferably even as part of the early bidding process.

The completeness of the set of *initial dependability requirements* is of course a major issue. Justifying completeness is always a difficult task; an impossible one, one could even argue. However, the initial requirements alone define the dependability properties that are expected from the system[5]. If there is no agreement on the completeness of this initial set, there is no definition of the system dependability to start with.

Besides, initial requirements delineate the scope of the dependability justification and case. A set of initial requirements – agreed to be appropriate and complete for the purpose – is thus an essential and necessary basis to define what

[5] Recently, B. Boehm wrote [10] that when he was working on the NASA High-dependability Computing Program with V. Basili and others, they found that one of their biggest challenges was just defining "dependability". Here, the *initial requirements* are the definition.

the system dependability is and to plan the time and resource needed to complete the case.

One important point to keep in mind is that completeness is a model property; the only way to establish completeness is by reference to a model. In particular to specify the **CLM0** requirements, a model of the system-environment interface is needed to precisely identify the environment states, events and properties of interest. Completeness of the set of **CLM0** specifications can then be established by comparison to this model. Completeness – and the need for precise models in safety justification – is discussed in detail in the descriptive part (Chapters 8 and 9).

Thus, the purpose of our work is to show how to demonstrate that the **CLM0** requirements are satisfied at all lower levels of the computer design and implementation. We shall see later that, in many cases, they can be formulated and justified independently from one another, but that arguments and evidence to support them may have much in common and may then be re-used.

5.3 The Other Foundation: The System Input-Output Preliminary Requirements

The first level of the dependability justification, the level-0, could be compared to the exoskeleton of molluscs such as clams, oysters, snails and many others, which is made up of hard elements totally different from the animal internal structure. Level-0 consists of "hard" application-specific **CLM0** requirements in essence totally different from the computer structures which must implement them. As we said in Section 5.1, the initial requirements do not even assume the use of a computer system.

A huge conceptual gap therefore exists between level-0 and the inner levels of the computer system implementation. This gap is another major difficulty of the justification that the **CLM0** specifications alone do not bridge.

Another level-0 foundation is needed to start the justification, namely the specification of "what the system is expected to do in the environment". That is, a preliminary specification of the relations that the system – viewed as a black box – is expected to maintain in the environment in order to satisfy the requirements, to deal with the effects of the postulated initiating events and comply with the environment constraints. These *system input-output specifications*, as we shall call them, play a pivotal role between the **CLM0** *requirements* and the lower levels of the computer implementation. They are qualified as *preliminary* because at this stage of the justification they are not yet validated; their validation against the *dependability requirements* will be the first step of the justification.

The **CLM0** *requirements* have to be mapped onto functions and properties of a computer system. At this early stage, these functions and properties – for example the functions implementing properties such as reliability, availability, and fail-safeness – cannot yet be precisely specified, nor do they need to be. The role of the computer system that must be assumed and described at this first level is that of creating and maintaining new relations in the environment compatible with the environment constraints.

As an example, the main preliminary *system input-output requirements* which address the replacement constraints and the use of the digital technology for the SIP system replacement described in Appendix A are:

fct.clm0: The SIP I/O functional relations must conform to the original logic

rel.clm0: The SIP global reliability and availability must remain at least as good as the original one

cmf.clm0: The SIP software common mode failure (CMF) level must not be superior to the CMF level of the original SIP hardware being replaced

sgfailure.clm0: No single failure should cause the loss of a protection function

autodetect.clm0: The SIP must detect its own failures (coverages achieved by auto-tests and periodic tests must be complementary)

failsafe.clm0: The SIP must be fail-safe

Computer system experts, together with application experts should take part in the elaboration of these preliminary specifications. Section 9.4.6 and the following ones in Chapter 9 discuss in detail a model for documenting *system input-output specifications*. The model takes into account the central and interlinking role of these specifications and is adequate for their validation against the **CLM0** specifications. Section 9.4.7, in particular, shows how these specifications can be expressed in terms of:

- A relation between system monitored and controlled variables,
- Properties on the domain and range of this relation,

and how this relation, domain, and range can be validated, *i.e.* shown to *satisfy* the **CLM0** specifications.

5.4 Primary Claims

Thus, the **CLM0** *requirements* and the *system input-output preliminary specifications* constitute the initial material which is given and on which the dependability justification has to be built.

The evidence that this material – as it is given and documented – is correct, complete and implementable is the primary evidence required for the justification. Clearly, not all this evidence is available at this stage. Part of the evidence can be provided at this stage by the documented interface between the environment and the system. The other part is to be provided by the system architecture, design and modes of operation. Nevertheless, all the evidence needed must be either provided or specified and claimed at this stage. This is the role of what we call – at level-1 of the justification – the *primary claims*. In the further stages of the justification, some of these *primary claims* have to be reified into more specific *subclaims* on properties of the system implementation.

More precisely, the primary evidence needed is of two kinds. The first kind is evidence that supports claims on the validity of the **CLM0** documented environment system interface; typically, evidence of:

- The validity of the documented environment **CLM0** constraints, initiating events and their consequences,
- The correctness, completeness and feasibility with respect to the environment constraints of the *system input-output preliminary specifications,*
- The detectability and controllability of the initiating events.

Evidence for these properties must be available at this stage and is typically level-1 evidence. Level-1 *primary claims* requiring this evidence will be referred to as *primary validity claims.* For instance, the first SIP *system input-output requirement* **fct.clm0** mentioned in the previous section, which requires evidence of the validity of the SIP input-output specifications, has to be expanded into level-1 *validity primary claims*.

Evidence of the second kind addresses properties of the implementation of the *system input-output* specifications; typically, the following characteristic properties of the system implementation must be specified as precisely as possible and claimed:

- Correctness, completeness and accuracy of the implementation,
- Fail-safeness, diversity, dependability.

This evidence often is not yet available at this stage of the justification and is anyway not level-1 evidence. These *primary claims* of the second kind, referred to as the *implementation primary claims*, have to be subsequently expanded and reified into more specific *subclaims* requiring the proper evidence at the lower levels of the system implementation. For instance, the last five SIP *system input-output requirements* of the previous section have to be expanded into level-1 *implementation primary claims.*

The elicitation and formulation of level-1 *primary claims* are the initial steps which belong to the level-1 of the dependability justification; a more detailed example of these claims will be given in Chapter 9 and can be seen on Figure 9.2, page 147.

These essential founding and constituting tasks of the dependability justification require creativity and expert judgment, but we will show how they can be guided by some principles. Those are some of the most important issues analysed in Chapters 0 and 9.

5.5 Differences Between Dependability Requirements and Claims

At this point, we should clarify the reasons why we find it necessary to make a distinction between “claims” and “requirements”, although by some aspects the two notions may appear quite close to each other.

In some respect, indeed, claims are like system and design requirements. They all address properties of the dynamic behaviour of the system implementation and/or properties of the interactions of the implementation with the environment. But a first difference is that, while requirements specify what the system should do and how, claims specify the evidence required (i) for justifying that what must be done is indeed done and (ii) the way it is done.

System architects and designers use the terms "requirement" and "requirement specification" in relation to their design work. They do not normally talk about "claims". One usually starts to make claims on a computer system when an application has to be made for approving or licensing the system for a given usage, or when dealing with regulators, safety authorities or their technical support organizations, or when submitting the system to independent assessment. In those circumstances, another difference is that a requirement is essentially a statement specifying what the computer system is originally designed to do and how, while a claim is a statement addressing the appropriateness of these specifications for a given usage.

To make the relationship between claim and requirement more precise, we can view a claim as being either a *meta-requirement* or an *extra-requirement*.

A claim can be viewed as a *meta-requirement* when it is a statement requesting evidence of a property that a (set of) computer system requirement specification(s) should enjoy: *e.g.* validity, consistency, completeness, correct implementation, or maintainability in operation.

A claim can be viewed as an *extra-requirement* when it is a statement requesting evidence of a property or a function that was not explicitly part of the original computer system requirement specifications; such an *extra-requirement* can be a *functional or a non-functional claim*, depending on whether it claims the existence of an extra functionality or an additional property. For instance, one may have to claim that a monitoring system is fit for operations not only in normal operation but also in post-accidental conditions. It must then be claimed and shown to satisfy some design criteria – *e.g.* a single failure criterion or a maximum reaction time – required for this specific mode of operation, although the system or some of its components may not have been originally specified and designed for this purpose.

Claims and requirements may coincide in contents in the ideal situation when the dependability justification is part of a development project and progresses along with the specifications and the design of the system.

Components off the shelf (COTS) and other pre-existing components used in systems important to safety are a typical case where a distinction between requirements and claims is useful. Claims that need to be made on the use of the component usually do not coincide with the requirement specifications according to which the component was designed.

A third and more subtle difference is that design requirement specifications focus on system functionality while dependability *claims* need to also comprehensively cover failure modes and undesired events like threats or hazards external to the system. For example, a functional specification for a monitoring system may be "to display the pressurizer coolant temperature and level values", while a functional dependability claim may in addition require evidence that silent

failure modes are excluded and in this example demand evidence that "pressurizer coolant temperature and level values be displayed *if validated measurements are available and not otherwise*". Thus, a *computer system* may satisfy its functional requirement specifications, and yet not necessarily satisfy a functional dependability claim.

Other examples of claims illustrating these differences can be found in Section 4 of [14].

To sum up, a claim is a statement that needs justification and requests evidence of either:

- An additional functionality or a property that must be enjoyed by the system (*extra-requirements*),

or

- A property that must be enjoyed by (a set of) requirement(s): validity of a specification, completeness, correctness of an implementation, maintainability (*meta-requirements*).

5.6 Evidence and Model Assumptions.

A clear distinction must be made between *"what"* to demonstrate (for instance the satisfaction of the *primary claims* and the *initial dependability requirements*) and "*how"* to demonstrate (in terms of *evidence and arguments).* So far, we have dealt with "*what"* only. Let us now pay attention to *"how"* which, obviously, is our main concern in this book. What is evidence? A complex question! If "to be believed, a story has to be true", the counterpart also carries truth: "how could a story be true if there is no way to believe it ?"

Evidence and assumptions are the two pillars on which rest our beliefs.

Evidence is what makes a claim assertion true in the model in which the claim is formulated. Formally, in mathematical logic, evidence consists of valuations which make predicates true when they are interpreted in a given model, essentially a mathematical structure and its universe. These concepts, the formal analysis of evidence and its properties are essential in dependability justification and are the major subjects of Chapter 8.

In practice, the evidence material of a dependability justification or a safety case consists of factual data that can be of different types. They may be documented results obtained by real tests or measurements, simulation runs, software static analyses, formal proofs; or feedback data of documented operational experience; or data establishing properties of a hardware component, of an actuator, of a sensing device; or the establishing of the existence of periodic testing or re-calibration procedures, alarms, manual controls and by-pass, *etc*.

In industrial safety cases, evidence is often erroneously confused with assumptions. Assumptions are inherent parts of the models that are used to formulate and to interpret claims and evidence. Chapter 8 helps make this distinction clear and analyses its implications, while Chapter 9 is an illustration of how in practice one can deal with these implications. The confusion obviously can

be dangerous since assumptions are taken for granted and not justified. For instance, the assessment of a safety critical software system that would implicitly "assume", without further ado, a perfect hardware, a given range of environment conditions, or a given behaviour of sensing devices can obviously lead to catastrophe.

But assumptions, of course, cannot be dispensed with. A dependability justification has to start somewhere without re-questioning the whole real world every time. Assumptions and the models based on them need thereby to be carefully and explicitly made. This important issue is also dealt with in Chapters 8 and 9.

5.7 How to Organize Evidence? A Four-level Structure

The evidence required for the justification of a single dependability property of a computer system is well-known to be of a quite disparate nature. B. Littlewood and his colleagues [29, 60, 62] must be credited for being amongst the first who recognised and seriously attacked this problem. The difficulties raised by this disparity are exacerbated by the drastic discontinuity between the environmental universe to which the properties to be justified belong and the computer system inner universe where the evidence is to be found. Remember the exoskeleton of molluscs metaphor at the beginning of Section 5.3.

The very first objective of this study was to find an organization for the heterogeneous sources of evidence which would make the justification of a dependability property of the "exoskeleton" easier and more convincing. The most obvious and natural direction to take first was to look at structures which had proved to be the most efficient to specify, design and implement the hardware and software of a complex computer system, and to remember that a well-known principle of engineering is that these structures are necessarily hierarchical (*cf. e.g.* [12, 73, 74, 93, 94]).

Practical experience, accumulated in several safety cases in nuclear engineering, later corroborated by a formal approach [14], suggested that the evidence required to justify *an initial dependability requirement* can be organized in a multi-level structure.

This key observation was fundamental. In all cases, the justification of a correct implementation of a requirement for prevention or mitigation of a hazard or a threat at the plant – computer system interface always appeared to require convincing *evidence of one or more different types*:

- Evidence that the functional and/or non-functional specifications of how the computer system must deal with the hazard/threat are valid, *i.e.* are sound, consistent, complete;
- Evidence that the computer system architecture and design correctly implement these specifications,
- Evidence that the specifications remain valid and correctly implemented in operation and through maintenance and operator control interventions.

In all cases met in practice, the required evidence always consisted of facts and data of one or more of these types. More precisely, any *dependability initial requirement* had to be supported by evidence material at one or more of the four levels:

- *The environment-system interface (system functions and properties expected by the environment)*: evidence of the adequacy of the set of functional and non-functional system specifications to satisfy the dependability requirement and to deal with the environment-system constraints and undesired events;
- *The system architecture:* evidence that the computer system architecture satisfies the functional and non-functional system specifications;
- *The system design*: evidence that the system is designed and implemented so as to perform according to the functional and non-functional system specifications;
- *The control of operation*: evidence that the dependability requirement remains satisfied during the system lifetime, given the environmental constraints and the way the system is integrated and operated in its environment. This includes evidence that the system does not display behaviours unforeseen by the specifications, *i.e.* that behaviours outside the design basis will be detected, controlled, and their consequences mitigated.

These levels are *not arbitrarily* chosen. We will see in Sections 5.8 and 6.3 – and in greater depth in later chapters – how these four levels are necessary and sufficient to construct arguments that relate disparate evidence material from the four levels to justify an initial dependability requirement. In particular, Section 8.12 addresses the more fundamental question of how many levels a formal justification structure should have in general, and how refined these levels should be.

But, from a practical viewpoint, the following considerations are sufficient at this stage. The first and fourth levels are usually easily understood and accepted by experts as being indispensable sources of evidence, although claims for evidence of the fourth type are often not explicit in industrial safety cases.

On the other hand, the necessity of a distinction between an architecture level and a design level may appear less obvious to computer and software engineers for whom the term architecture may take on different meanings; in particular, they often consider the processor architecture (*e.g.* its set of instructions) and the software architecture (*e.g.* its modular organization) as integral parts of the design.

For the justification of dependability, however, it is essential to distinguish between the architecture of the system which is intended to respond to certain failures, and the system internal design. Failures caused by undesired events occurring in the environment may require redundancies and diversities of functions, components and communication lines that dictate specific aspects of the system architecture. Besides, as Peter Neumann pointed out in his ACM "Inside Risks" column [71], specific modes of failure may result from inadequate attention to what he also refers to as being the architecture of the system. Distributed control over networks of redundant or diverse unreliable subsystems and components, with distributed sensors and distributed actuators is particularly

vulnerable to widespread outages and perverse failure modes such as deadlocks, race conditions, starvation and other unwanted interdependencies. It is at the level of the system architecture that specific defences must be conceived to detect and tolerate these failures and not only in the design of isolated components. While design correctness is a property of components, reliability, survivability, fault - tolerance, security are properties of component interactions for which architecture is an indispensable source of evidence.

Hence, for our purposes and in the context of dependability, we adopt the following definition:

Definition 5.2 We regard the *system architecture* at level-2 as being the organization in space and in time of the dependable hardware and software components of the system, and of their control and communication interfaces.

To make things more concrete, let us give some examples of evidence material for each level:

- At level 1: a regulation or a regulatory position asserting or justifying the validity of a given system requirement specification; a safety analysis report.
- At level 2: a statement from a certification body guaranteeing certain properties of a hardware/software platform; a (set of) test result(s); the proof of the existence of an architectural hardware or software property such as redundancy or diversity.
- At level 3: the coverage of a (set of) module or integrated test result(s); a conclusion of a code static analysis or failure mode analysis; existence and proof of certain applied defensive programming measures.
- At level 4: existence proof of an invariant property or a system state guaranteed by periodic tests, by operator control interventions or by operational procedures; a (set of) pertinent operational feedback data.

At any of these levels, a same element of evidence can of course be used to support more than one initial dependability requirement.

Other examples of different types of evidence are given in Table B.2 of Appendix B for the justification of a claim on the safe use of a nuclear material handling system.

Evidence material at the four levels must consist of facts and data, taken for granted and agreed upon by all parties involved in the dependability case. There must be confidence beyond any reasonable doubt in the truth of these facts and data, without further evaluation, quantification or demonstration. In order to consider the evidence as being unquestionable, this confidence must be supported by a *consensus* amongst all protagonists. Besides, to be validated, we will see that evidence must be amenable to *plausible* models. *Consensus* and *plausibility* are essential concepts discussed in greater detail Section 7.2.

5.8 How Are the Four Levels Related? Levels of Causality

Hence, four different levels of evidence, each level consisting of possibly various evidence elements, support an *initial dependability requirement*. Clearly, the difference between these levels is not degree, value or priority, but the intrinsic nature of evidence. How, then, can these levels be mutually related so as to build a coherent argument justifying an *initial dependability requirement*?

The four levels can be viewed as four *levels of causality*. To each level correspond specific system properties and functions and also undesired events likely to cause these functions to fail or these properties to be invalidated. A function may rely on properties or functions implemented at lower levels and may thus also fail as a consequence of undesired events occurring at these lower implementation levels. Similarly, a property may rest upon and imply the existence of lower level properties. The consequences of undesired events are therefore dealt with at each level on the basis – more precisely on the "implication" – that consequences of lower level undesired events are (or will be) properly dealt with at these lower levels. Such an implication is similar to claiming at each level that lower levels are somehow free of unsafe failures. Similar implications are also inherent to hierarchical design processes.

Because of these implications a claim on the dependability of a function or a property at a given level usually implies claims on evidence of lower levels. And because evidence material is provided at four different levels, any *initial dependability requirement* or *primary claim* may need to be expanded into claims for evidence at any of the lower levels.

In Section 5.4, we explained that, at the first level of the environment-system interface an *initial dependability requirement* needs to be expanded into a set of *primary claims* of two kinds.

Primary validity claims address properties of the documented environment-system interface; they claim the validity of this documented interface. These claims are entirely supported by level-1 evidence. But the other *primary claims* require evidence of a correct and fail-safe implementation of the system. These claims must be supported by evidence at the lower levels. At level-1 these *primary implementation claims* actually infer properties of the architecture, design or control of operations. They are assumed to be satisfied at level-1 but need to be justified by evidence at lower levels.

At level-2, the architecture level, each *primary implementation claim* not supported by evidence at level-1 and assuming properties of the architecture must be expanded into level-2 claims. As an example, a *primary implementation claim* may imply the correct supervision of the system by an independent watchdog. This *primary claim* would imply at level-2 a claim asserting that:

(5.1) **watchdog.clm2:**

(system is in initstate and watchdog is disarmed)
or (system is not in initstate **and** watchdog is armed)

Table 5.1. Dependability Justification Level Organization

Justification and Evidence Level:	Requirements/ Claims:	Inferred from evidence:	Expanded into and inferred from delegation subclaims at lower levels
0	Initial Dependability Requirement	(Out of justification scope)	Plant–System Interface Primary Claims
1	Plant-system Interface Primary Claim	Plant-system interface Evidence	Architecture Subclaims
2	Architecture Subclaim	Architecture Evidence	Design Subclaims
3	Design Subclaim	Design Evidence	Operation Subclaims
4	Operation Subclaim	Operation Evidence	(Out of justification scope)

In turn such a level-2 claim must be supported by level-2 evidence and/or expanded into lower level claim; and so on, down to the lowest level of operation control. For instance, it may be implied at the architecture level that output channels unfailingly detect data that are invalid before delivering them to voting mechanisms or actuators; this implication requires at level-3 a claim asserting that:

(5.2) **output_channel_register.clm3**:

$$\forall t\text{: (register value}(t)\text{ is in range } \underline{\textbf{and}}\text{ validbit}(t) = 1)$$
$$\underline{\textbf{or}}\text{ (valid bit}(t) = 0)$$

Every *initial dependability requirement* is therefore the root node of a tree structure of nested subclaims at one or more of four levels: the environment-system interface and the system architecture, design and control of operation.

The four levels are *hierarchical levels of causality* because claims, properties and functions at each level i, i = 1...4, depend upon claims, properties and functions at the lower levels $j > i$, and not upon those of the upper levels. The reasons for this one-way and top-down dependency are to be found in the hierarchical design organizations of complex computer-based systems; these reasons are re-examined at the light of the formal developments of Chapter 8 in Sections 8.12 and 8.13. Let us just recall here that the top-down stepwise refinement of design abstractions, the robustness, adaptability and efficiency of the behaviour of hierarchical organizations and the absence of cycles of dependencies are some of the main factors which naturally lead to hierarchical structures, both in engineering and in the natural world; key references on this subject are [1, 12, 24, 74, 94].

This hierarchical dependency is clearly also beneficial for the justification of claims on system properties since it precludes the existence of loops in arguments

supporting these claims. This is our first observation in this book – and not the last one – of a system property being essential to both design and to validation, leading to the presumption that only well-designed systems can be validated and conversely.
It is useful at this point to summarize and capture by the following definitions the essential notions of the four-level structure introduced so far:

Definition 5.3 A *claim* stated at some level i, , $i = 1...4$, is a statement that requires *evidence* from some lower level j, $j \geq i$. Claims are referenced by a unique identifier that specifies its name and its level: **<name>.clm** i; a *claim* can also be qualified as being either a *primary claim* if $i = 1$ or a *subclaim* if $i = 2...4$.

A more precise definition of a subclaim will be given in Section 6.3.

Definition 5.4 *Evidence* at some level i, $i = 1...4$ is factual data that make true a *claim* statement **<name>.clm** i. An element of *evidence* at level i is referenced by an identifier that specifies its name and its level: **<name>.evd** i, $i = 1…4$.

While the notion of *claim* is syntactic, *evidence* is part of the semantics of the justification. How evidence makes a *claim* statement "true" requires a model to formulate and valuate the statement, which will be the subject of the Description Part of this book.

We can now also define what we mean by *dependability justification*:

Definition 5.5 A *dependability justification* is the set of *claims* and *evidence* components which justify the implementation and operation of a set of *initial dependability requirements* on a computer system, given the environment constraints and the preliminary input-output specifications of this system.

A *dependability justification* is an essential part of what is often called in industrial practice a *safety case* or more generally a *dependability case*; in line with nuclear regulators' guidelines [28] and other authors [66], we can define :

Definition 5.6 A *safety* or *dependability case* is the set of activities to be undertaken in order to carry out and document a *dependability justification*, including the verification and validation activities, the organizational arrangements to ensure the competence and independence of those undertaking the justification activities, to ensure the legal negotiations and compliance with the applicable standards, guidelines and regulations, and the setting up of a feasible programme for completion of these activities.

If we were to venture a comparison with linguistics, *dependability requirements* and *claims* would be the syntactic elements of the *justification*, *evidence* would be semantics elements, and the *safety case* would cover the pragmatics and the socio-linguistic aspects; the latter aspects, albeit important, are only alluded to in this work.

Table 5.1 gives an overview of the four-level-organization, which relates dependability requirements, primary claims, subclaims and evidence.

5.9 Examples

These concepts can be illustrated by examples drawn from the safety case of the nuclear material automated storage, retrieval and handling vehicle described in Appendix B.

The *initial dependability requirement* considered in Appendix B specifies that the vehicle represented on Figure B.2 and its vertical lift must be deactivated while a pallet loaded with material is being inserted or removed horizontally from its cabinet.

The corresponding level-0 *dependability requirement*, already mentioned in section 5.2 is (requirement (B1), page 297):

(5.3) **no_displacmt.clm0**:
horizontal vehicle and vertical lift displacements are prevented when pallet is being inserted or extracted from a cabinet

And a *system input-output preliminary requirement* specifies what the logic of the vehicle control microprocessor is expected to do in terms of variables describing the state of different components of the vehicle:

(5.4) **<no_ displacmt_system specification>:**
if
(vehicle docked) **or** (vehicle-cabinet clamped) **or** (fork/shuttle engaged) **and** ((drive mode on) **or** (lift mode on))
then
<disable vehicle drive and lifting device motors within 0.1 sec>

At level-1, a *primary claim* must claim evidence for the validity of this system specification:

(5.5) **no_displacmt_spec_validity.clm1:**
specification (5.4) is valid

Part of the level-1 evidence material (plant–system interface), which supports this *primary validity claim* is to be found in the documented specifications of the vehicle (Table B.2, page 305):

AGV_spec.evd1:
Documented specification of the automated guided vehicle operation and interlocks

At level-2, the system architecture, a *subclaim* from which is inferred the *primary claim* for evidence of the correct implementation of the system specification (5.4), is (Table B.1, page 302):

(5.6) **inputch.clm2**:
set of AGV control unit sensing input channels is complete and adequate; status signals (docked/undocked, sealed/unsealed, fork/shuttle positions, drive mode) are correctly captured

This *claim* requires that states of vehicle drive, lift, fork, shuttle and cabinet clamps are correctly captured by corresponding sensing devices and channels.

One of the level 2 evidence documents supporting this *subclaim* is (Table B.2, page 305):

sens_spec.evd2:

documented specifications, performance and failure characteristics of input sensors

A *subclaim* at level-3 (design) from which is inferred the *primary claim* (5.4) for evidence of the correct implementation of the system specification is (Table B.1, page 303):

(5.7) **correct_code.clm3**:
the code implementation of the specification (5.4) is correct

A level-3 design evidence components supporting this *subclaim* is (Table B.2, page 306):

code_utst.evd3:
reports of code unit tests

One of the *subclaims* at level-4 (control of operation) from which are inferred level-2 *claims* for evidence of a fail-safe architecture, is (Table B.1, page 304):

(5.8) **oper_fma.clm4**:
adequate anticipation of failure modes of vehicle, fork/shuttle, sensors, actuators, power supplies, motor lock

Claim (5.8) requires a pertinent return of experience from past operation to ensure that all potential failure modes of the implementation have been anticipated. Level-4 operational evidence material supporting this subclaim oper_fma.clm4 is (Table B.2, page 306):

opr_rex.evd4:
data from probation period report, operators reports, operational feedback from other similar AVG's

The detailed expansion of the justification of the dependability requirement **no_displacmt.clm0** is given in Appendix B.

http://www.springer.com/978-1-84800-371-2

Justifying the Dependability of Computer-based Systems
With Applications in Nuclear Engineering
Courtois, P.-J.
2008, XVIII, 323 p. 24 illus., Hardcover
ISBN: 978-1-84800-371-2