Semantic Structures and Logic Properties of Computer-based System Dependability Cases

Pierre-Jacques Courtois, *Member IEEE*

Université de Louvain-la-Neuve and AV Nuclear, Belgium

courtois@info.ucl.ac.be

November 1998

(A revised version appeared in *Nuclear Engineering and Design*, 203 (2001) 87-106)

Contents

Abstract: This work addresses the issue of structuring the validation process of dependable computer based systems. It was motivated by the desire to make the licensing and certification of these systems more reliable and efficient. The paper analyses the structural, semantic and logic properties of the demonstration that a computer based system is adequately specified, designed and maintained in operations. Three classes of dependability claims are identified; those which address the environment - system interface, the design and the operational behaviour. A structure is proposed to analyse the relations between these classes and the convergence of their supporting evidence. Relations and formal properties which should be satisfied by the underlying models, by the languages required for the interpretations of the real domains, and by the proof obligations are identified. The implications of these properties on design criteria and design mechanisms such as reflection are discussed. A real industrial dependability case is used to illustrate the concepts and discuss their applicability.

Key words - Safety critical software, safety assessment, dependability case, combining evidence, model theory, formal model, interpretations.

1 Introduction

Computers are more and more used to implement functions where safety, availability, reliability, or more generally dependability is at stake. Such qualities remain sterile if they cannot be predicted and demonstrated. It is not sufficient that a system be safe and its software correct; there must also exist a means to know it.

The demonstration that a system is dependable for a given application is quite hard, and often responsible for unexpected licensing costs and delays. This is apparently true of any complex artifact even when no computer is involved. The crude approach followed in practice is to collect all possibly available information about the application, the system, and the design, and to base a final approval on the combined evidence found therein.

The work reported here was prompted by difficulties met with this brute force approach over the last six years in a number of industrial safety cases involving nuclear safety critical software.

The following three paragraphs briefly discuss what are believed to be three root causes of these difficulties and why they motivated this work, care being taken to avoid bias which may be induced by the specific aspects of nuclear applications:

1) *Environment - System Interface*.
Many dependability requirements - for example those concerning safety or security - are originally determined by the application in which the computer and its software are embedded. What is required from the computer implementation is essentially to be an available and/or reliable implementation of those requirements. On the other hand, many pertinent arguments to demonstrate dependability - for instance the provision of safe states and failure modes - are not provided by the computer system design and implementation, but are determined by the environment and the role the system is expected to play in it. Previous attempts at structuring the safety cases of computer based systems have concentrate on the V&V problems raised by the computer and software technology and have paid too little attention to this aspect.

2) *Life-Time Coverage.*
Dependability depends not only on the design, but also, and ultimately, on the installation of the integrated system, on operational procedures, on procedures for (re)calibration, (re)configuration, maintenance, even for decommissioning in some cases. A dependability case is not closed before the actual behaviour of the system in real conditions of operations has been found acceptable. The demonstration of the dependability of a software based system therefore involves more than a code correctness proof. It involves a large number of various claims spanning the whole system life, and well-known to application engineers, but often not sufficiently taken into account in the computer system design.

3) *Model Interpretations*.

Dependability - although its absence may have severe real consequences in terms of life losses and material damages - remains somehow an immaterial property. The demonstration of its existence - especially for complex and digital systems - cannot be strictly experimental and obtained by e.g. testing or operational experience. For instance, a dependability case does not only include claims of the type:"the class X of unacceptable events shall occur less than once per Y hours in operation". It also includes or subsumes the claim that the class of events X is adequately identified, i.e. is complete and consistent. Dependability can only be discussed and shown to exist as *abstract properties of models of the system behaviour, of its interactions with the environment and of accident scenarii.* These models must be unambiguously understood and agreed upon by all responsible actors of a dependability case: users, designers and assessors. This is unfortunately seldom the case. Claims for dependability - although usually based on a huge engineering and industrial past experience - often remain only informally specified. Application domains, user interfaces, and industrial environments often are also seldom or poorly formalised. One symptomatic and unfortunate consequence of this situation is the deplorable tendency in licensing negotiations to "juggle with assumptions", i.e. to argue and interpret system or environment behaviour or accident hypotheses in order to claim increased levels of safety or reliability without making any change to the system itself or to the environment. A great need exists for more formal treatments in industrial dependability cases, i. e. for more reliable models and methods of interpretation.

Based on these observations, our approach is an attempt to provide a structure to organise the variety of different claims of a dependability case, to avoid or minimise misinterpretations, and to more systematically and formally establish the convergence of the various pieces of evidence which support these claims.

Basic notions of model theory (see e.g. [Bell 1977]) were found useful to address these structural and formal aspects of the validation process. A branch of mathematical logic, model theory analyses the relationship between sets of first-order sentences and the structures in which they are satisfied, i.e. their models. It is, in particular, concerned with the methods by which models with prescribed properties can be constructed.

2 Case Example

We shall use the real safety case of a nuclear monitoring computer based system to illustrate the notions and concepts introduced in the course of the following sections, and to discuss their applicability. This real case is one of those which are modelled by a bayesian net in [Courtois 1998].

After the Three Mile Island accident, nuclear utilities revised their accident procedures in order to cover larger domains of abnormal or accidental plant states [NUREG 0696]. Computer based diagnostic systems were designed to monitor these critical state domains. Among the various functions expected from such a system the one we have selected for our example is the inventory of the reactor coolant for heat removal. This function has to

monitor the coolant levels in the pressurizer and in the in reactor vessel, and the margin to saturation. When the reactor existing instrumentation does not permit direct measurement of the coolant level in the vessel, a special algorithm using other data must be designed to infer and assess this level. This assessment must take into account the possibility that - under special circumstances - steam may accumulate under the vessel head. The possible presence of such a steam bubble must therefore be detected and announced to the operator. For the particular reactor we are concerned with, the conditions to be monitored for this detection were presumed to be associated with a rapid cooling of the coolant while being under natural circulation, together with a rapid increase of the pressurizer level.

It is interesting to note at this point that the above description is not much different in form and contents from the highest level specification of the monitoring function which was initially documented by plants engineers for system designers.

3 Claims and Evidence

What is the most often -overlooked risk in software engineering?
That the environment will do something the designer never anticipated.
James J. Horning, ACM SEN 23, 4, 1998

In [Bloomfield 1995], a *safety case* is defined as "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment". We want to extend this concept to the concept of *dependability case*, which may cover other issues than safety, such as security and availability.

We regard a dependability case as *evidence that a given set of dependability claims are satisfied by a computer based system* (for short referred to as the *system*) *in a given environment.* This set of claims is assumed to be consistent and complete. How to define this set of high level requirements is an issue which must be resolved by safety and engineering experts of the domain of application, and which is considered as being outside the scope of this paper. What we shall be concerned with is to ensure that each claim of the set is satisfied by the implementation.

We make the distinction between the notions of *claim* and *requirement*. Dependability claims are viewed as required properties of the dynamic behaviour of a system implementation, and of its interactions with the environment. When these properties are explicitly specified by the functional and non-functional requirement specifications of the system, then claims and system functional or non-functional requirement specifications coincide. There are situations, however, where the properties *claimed* from the system behaviour and from its interactions with the environment neither are a part, nor are an immediate consequence of the system requirement or design specifications. For instance, it is reasonable to claim that our monitoring system is fit for operations in post accidental conditions and that it satisfies the design criteria - e.g. the single failure criterion and a maximum reaction time - required for these operations, although some of the system components may not have been originally specified and designed for this purpose. COTS

components used in systems important to safety are another case where dependability claims usually do not coincide with the system specifications.
Another difference is that system requirements and design specifications focus on system fonctionality while dependability claims focus on system hazards [Wong 1998]. For example, a system specification for our monitoring system is "to display the pressurizer coolant temperature and level values". In contrast a dependability claim states that "pressurizer coolant temperature and level values must be displayed if and only if validated measurements are available". A system may satisfy the system specification, while failing to satisfy the dependability claim.

A clear distinction must also be made between *what* to demonstrate (the *dependability claims*) and *how* to demonstrate (the *evidence*). Practical experience, also supported by the formal approach discussed later, has shown that in general *an application dependability claim and its supporting evidence can be organised in a three-dimensional structure.*
This observation - fundamental to this work - is based on the fact that a claim for prevention or for mitigation of a hazard at the application level is necessarily composed of claims - we shall call them *subclaims* - which are of three types: subclaims of adequate functional and/or non-functional design specifications to deal with the hazard, subclaims of correct design and implementation of these specifications, and subclaims that these specifications are maintained in operation. The supporting evidence for a dependability claim made at the application level can be organised along the same structure.

Thus, every application-dependent *dependability claim* and its supporting evidence consists of subclaims and evidence components along the following three *dimensions:*

1. *Environment-system interface*: *subclaims and evidence* of the adequacy of the set of functional and non-functional requirement and design specifications to satisfy the dependability claim and deal with the environment/system constraints and exceptions;

2. *Design*: *subclaims and evidence* that the embedded computer system is designed and implemented so as to perform according to the above specifications;

3. *Operations-environment interface*: *subclaims and evidence* that the dependability claim and the environmental constraints will remain satisfied during the whole life-time of the computer based system - in the way it is integrated and *operated in its environment.* This includes evidence that this system does not display behaviours unanticipated by the specifications, or that such behaviours outside the design basis will be detected and their consequences mitigated.

A dependability claim for the monitoring system of our example is expanded into subclaims along these three dimensions on figure 1.

Environment - system subclaims and their supporting evidence components pertain to the adequacy of the specification of the algorithm chosen to detect the pressurizer abnormal behaviour - by observing the coolant in core temperature T and the pressurizer coolant level L - and of the specifications of the actions to be taken in the presence of out of range data or failure of the computer and communication system.

The demonstration that these system specifications are "right", i.e. that they correspond to the intention of the dependability claim, is an essential and usually hard part of a dependability case. In our example it turned out to be one of the most difficult step - currently not yet fully resolved. The corresponding supporting evidence - see figure 2 - depends in this example on the predictions of plant simulation models, the results of a failure mode analysis, and the competence of plants engineers.

Dependability Claim:	**Abnormal pressurizer behaviour is detected and signalled to operators within 6 minutes**
Environment-System Subclaims:	Adequacy of parameter ranges, unit conversion, accuracy. Adequacy of the system specification: if [-(Δ**T**/Δt) > 5°C/h and (Δ**L**/Δt >5%/min) and (3 primary pumps stopped)] then <send alarm>; Adequate specifications of detection mechanisms and fail-states for **T** and **L** out of range values, data link failures, and computer system failures. .
Design Subclaims	Adequate A/D input conversion, number representation and algorithms for temperature (**T**) and water level (**L**) gradients. Adequate auto-tests, exception and error procedures Correct executable code. Correct hardware implementation.
Operations-Environmentl Subclaims:	Adequate and Robust (e.g. to operators' errors) in-service Procedures for **T** and **L** (re-)calibration, periodic testing, maintenance. Adequate anticipation of failure modes of computer hardware, sensors, power supplies and plant interfacing instrumentation. Absence of spurious alarms in operations.

Figure 1: Structure of subclaims of the dependability claim: pressurizer abnormal behaviour detection. **T**: coolant in-core temperature; **L**: coolant level in pressurizer.

Design subclaims and evidence components are similarly summarised in figure 1 and 2 respectively. They support a correct integrated hardware and software implementation of the specifications.

Dependability Claim:	**Abnormal behaviour of the pressurizer is detected and signalled to operators within 6 minutes**.
Environment and system evidence:	- Predictions from thermo-dynamic plant behaviour computer model. - Competence and past experience of plant engineers. - FMECA report and operational feedback from other plant incidents.
Design Evidence:	- Quality of specifications. - Formal or mechanical correctness proofs. Results of code unit tests. - Competence and past experience of programmers, of supplier of instrumentation.
Operation-environment Evidence:	- Operational experience on thermocouple and gauges equipment; - Integrated on site tests; - Conclusions from probation period, operator reports, other similar installations.

Figure 2. Evidence components for dependability claim on pressurizer behaviour.

Operations - environment subclaims and evidence address the behaviour of the system while in operations in its real environment. In fine they form the most essential and conclusive part of a dependability case, especially when COTS components are involved. And yet they are almost always ignored by software engineers, inclined to believe that this third dimension is superfluous because these claims should be part of the system and design specifications. This dimension includes all the subclaims which cannot be established otherwise than by evidence that the integrated system operates correctly in every mode and in its real environment. For the monitoring system, operational evidence must for instance give assurance that operator in-service interventions such as (re)calibrations and periodic tests are feasible and do not cause spurious alarms; that there is no undesirable interference with existing instrumentation, that post-accidental conditions and failure modes have been comprehensively anticipated and lead to safe states.

Operational evidence is in part experimental, and obtained by environment simulation, integrated tests, probation periods or operational feedback. It is also the evidence that the the continuous monitoring functions and periodic tests of the system, while in operations, adequately anticipate and cover all undesirable behaviours. Such evidence can greatly contribute to the confidence one has in the system. Often, it can be obtained only after the system or its prototype has been installed and tried in real conditions.

This three dimension organisation is applicable to non-functional dependability claims. Figure 3 shows for instance the subclaims which correspond to the requirement that the monitoring system should be acceptable for post accidental conditions (PAM function).

Dependability Claim (non-functional):	Abnormal pressurizer behaviour detection & signalling is available as a PAM (post accidental monitoring) function.
Environment/ System Subclaims:	Adequate set of selected design criteria, namely : - quadruple HW redundancy; - physical isolation of I/O and of power supplies; segregation from non-safety functions; - single failure criterion; fail-safe behaviour of detection and signalling function;
Design Subclaims:	Fail-safe computations for input data validation / invalidation, causing no spurious alarms; Single failure network architecture and protocols;
Operations-environment Subclaims:	Adequate anticipation of impact of post-accidental conditions on the embedded system; Adequate anticipation of input devices and computer failure modes; Robust (re-)calibration, periodic test and resetting in-service procedures.

Figure 3. Non-functional dependability claim.

4 Convergence of Evidence

The term "dimension" which is used to introduce the three types of subclaims and evidence components is intended to convey the notion that a particular dependability claim at the application level and its supporting evidence can be viewed as two projections of subclaims and evidence components respectively on one or more of these three dimensions. Along each dimension, evidence can of course be of different nature - for instance deterministic or probabilistic. It is likely to be mainly deductive in the first two dimensions and empirical in the third one. Evidence can also offer different levels of confidence. The quantitative evaluation of the confidence achieved - for instance by probabilistic measures - is an important issue not directly addressed in this paper, but the model proposed could integrate this evaluation.

We argue that the three dimensions are both *necessary* (in terms of subclaims) *and sufficient* (in terms of evidence components).

They are necessary in the sense that *one cannot ignore subclaims* in any one dimension for a dependable application. This is rather obvious for environment/system and operational subclaims. The necessity of design subclaims, perhaps less obvious, is supported by the same arguments which require that not only the product, but also the development process needt to be assessed for dependable software based systems, or at least cannot be ignored without justifications.

Note that the dimensions are not mutually independent. A dependability claim is translated and expanded into system subclaims that certain system requirements specifications are

adequate. In turn, every system subclaim for the adequacy of a system requirement induces and is expanded into design and operational subclaims for, respectively, the correct implementation and maintenance of these specifications. This expansion process of (sub-)claims of one dimension into the next one will be formally analysed in section 7.

The three dimensions are also *sufficient* in the sense that a body of precise, pertinent and *converging components of evidence* in these dimensions is sufficient to claim a dependability property of a system behaviour. This notion of *converging components of evidence* implies the possibility that stronger evidence in one dimension may compensate for weaker evidence in another; for instance, system/environment or operational evidence that in operations a particular event never (or always) occurs can supplement a lack of evidence on the adequacy of the design or the implementation of a COTS component.

Every:	**is supported by components of :**	**is expanded into a set of:**
Dependability Claim	- - -	system subclaims
Environment-System subclaim	Environment evidence	design subclaims
Design subclaim	Design evidence	operational subclaims
Operations-Environment subclaim	Operational evidence	- - -

Figure 4: Structure of relations between subclaims and evidence components

Convergence of distinct components of evidence cannot be established without a structure in which the logical relations between these components and their respective roles to support the claims can be expressed. Figure 4 summarises the basic structure of these relations. Section 7 uses this structure as an axiomatic basis for a model in which the convergence of evidence components can be formally defined and established.

5 Interpretations

We observed in the introduction that dependability claims are abstract properties, necessarily formulated in some language, more or less formalised. They are necessarily related to the real implementation system and to its environment through models and interpretations of the real world. Besides, demonstration of dependability for complex digital systems cannot be strictly experimental. Central to a dependability case is a logical analysis based on mental representations of the real system and of its interactions with the environment.

The other main aspect of this work is therefore an attempt to give formal and logical foundations to a dependability case. The purpose is to clarify the linguistic and logical implications and to identify the nature of the relations that exist between the three

dimensions of subclaims and evidence. The intention is of course not to cover formally all practical details of an industrial dependability case, nor to provide methods for mechanical analysis. Only basic notions of *model theory* (see e.g. [Bell 1977]) shall be needed.

As said earlier, the issue of completeness and consistency of the set of dependability claims is application dependent and outside the scope of this paper. What, however, we are concerned are the means to ensure that:

- the translation of a dependability claim into subclaims in each dimension is "right", i.e. corresponds to what is intended and expected from a real system;
- the evidence components are "true", i.e. reflect what a real system and its environment are;
- the evidence components support the subclaims.

The first two issues must clearly be answered through a process of validation, as there is no higher level formal specifications against which to conduct a verification of the subclaims and evidence components. The understanding and the semantic aspects of the real system are primordial in this process.

The third issue is one of logic. If subclaims and evidence components are properly formalised in some system, the former could be logically derived from the latter.

The connection between "valid" and "provable" - as clearly explained in [Rushby 1993] - is of the same nature as between semantics and syntax. This connection, in model theory, is established by *interpretations* that associate a true or a false (informal) *statement* about some real world domain with each *formula* of a formal system. The purpose is to make the syntactical notions of theorem and proof coincide with the semantic notions of truth and model. If the axioms and assumptions correspond to true informal statements about the real domain - i.e. the system and its environment - then the theorems - the subclaims - that can be proven will also correspond to true informal statements about the real domain. And these true facts are deduced from given true facts simply - like correctness proofs - by following the rules of a formal system. No understanding of the real domain is required to carry out these syntactic operations.

The understanding of the real domain, however, is needed to select the non logical axioms and the assumptions, and to *interpret* the theorems (subclaims). Then, the difficulty - and the frustration when risk and dependability assessment are at stake - is that we cannot exhibit and directly deal with real entities. We are limited to use descriptions only. This fundamental limitation makes a "true" validation of *real* dependability claims impossible. Carefully defined *interpretations* of the real entity behaviours must be defined, and we must restrict ourselves to a demonstration - in terms of these interpretations - that the decomposition of an application dependability claim into distinct interrelated dimensions of subclaims and evidence components is "right", i.e. consistent and complete.

6 Structures

We have adopted the structural viewpoint that one can regard the real entities as being associated to (*mathematical*) *structures* (see e.g. [Bell 1977]). This choice - as opposed to a constructive or intuitionistic approach in which proofs would be based on feasible constructions rather than on structures - is not arbitrary. It reflects the fact that in a dependability case some structures, like some of those defined by environment or implementation constraints are given to us as existing "out there", finished and completed before we use them in our semantic analysis.

The notion of a *structure* is intended to be of the most elementary kind and of the most general applicability in order to capture all aspects of a real system with hopefully no - or a minimal set of - restrictive assumptions .

Informally, a *structure* merely consists of (1) a non-empty class called the universe or the *domain* of the *structure*, the members of which are the *individuals* of the *structure*, (2) possibly various *basic operations* and (3) various *basic relation*s on the domain.

As an example, elementary arithmetic can be defined as the study of a particular structure, that of individuals which are the natural numbers, basic operations being the addition and the multiplication, and the only relation being the identity relation. Set theory is also concerned with a structure whose individuals are all sets, identity and membership being the only relations. Elementary Euclidean geometry can be regarded as the study of a structure: the elementary Euclidean plane whose individuals are points and straight lines, with the properties (unary relations) of being a point, a straight line, and the ternary relation of collinearity.

For a dependability case we need descriptions of the three real domain entities involved, one in each dimension: the Hw-Sw implementation, its physical environment, its modes of use and operations.

Examples of *structures* which are relevant to the subclaims of the monitoring system (cfr. figure 1) are given in figure 5. The design structure can be regarded as a time - independent description of the computer system functionality, while the behavioural model captures the time-dependent interactions of the system with the environment (including the users).

D.L. Parnas's [Parnas et al. 1991] functional relations are a particular type of *structures* that are adequate for the description and interpretation of the design and -in part- environment dimensions. These relations provide a description of natural and environmental constraints (NAT) of system functional requirements (REQ), of system input-output (IO), of software specifications (SOFT).

Because dependability claims can be dealt with separately, the structures can differ from one claim to the other in a same dependability case.

7 Formal Model

"Le formel pur est stérile et sans intêrêt...
La valeur de la forme commence
lorsqu'un peu de matière la leste et la gauchit."
Michel Tournier, "Petites Proses" , 1986.

The notions introduced in the two previous sections indicate that a dependability case spans three distinct domains:

- A *real* domain **D** which consists of three entities: (i) the actual system (hardware and software) implementation, (ii) its physical and user environment, and (iii) its mode of use, operations and maintenance;
- The *semantic* domain of *interpretations*: *Interpretations* of the real entities, i.e. mappings between *structures* associated with these three entities and three languages $\mathbf{L_1}$ $\mathbf{L_2}$ and $\mathbf{L_3}$ respectively**.**
- The *formal* and *logical* domain: The syntax of the languages $\boldsymbol{L_i}$ and a logical system - for instance first order logic - used to formulate claims, statements of evidence, and rules of inference.

Dependability claim:	**Abnormal behaviour of the pressurizer detected and signalled to operators within 6 minutes; no automatic control action generated**.	
Environment-system structure:	*Domain:* *Basic Relations:*	- Set of postulated Initiating Events (PIE set); - Plants components: vessel, pressurizer; - P, T and L flow and thermodynamic equations; - PIE effects on T, L values
Design Structure	*Domain:* *Basic Relations:*	- States of sensors, of A/D converters, of input-output registers; - input data conversion and validation; - input-ouput register relations; - software functional relations.
Operations-environment Structure	*Domain:* *Basic Relations:*	- States of: sensors, calibration and test equipment, power supplies, other interfacing plant instrumentation, operator; - PIE set. - operator actions, plant procedures; - effects of: PIE's, accident scenarii, failures.

Figure 5: *Structures for pressurizer behaviour dependability claim.*

In practice, these three levels are always present in a dependability case. To make them properly coincide, however, is a major issue. Misinterpretations between the real domain

where failures actually occur, and the two other domains where dependability is specified and demonstrated are major causes of undetected faults and incidents.

Let us briefly recall some of the basic notions we shall need to deal with the formal and semantic domains.

The formal domain is needed because a ***proof*** pre-supposes the existence of a formal system or language **L** of *assumptions*, *axiom schemes* and *rules of inference*. The proof in **L** of a ***formula*** φ from a set of formulas Γ is then a finite sequence $\varphi_1,...,\varphi_n$ of formulas such that $\varphi_n = \varphi$, and where each formula is either an axiom, a formula from Γ, or else follows from earlier formulas by rules of inference. The formula φ is said to be ***provable*** from the formulas Γ (written: $\Gamma \vdash \varphi$).
A ***sentence*** ψ is a formula without free occurences of variables, i.e. where all variable occurences are bound by the scope of a universal quantifier naming that variable. A ***theory*** in **L** is a set of **L** sentences Γ which is closed under deducibility, i.e. such that for each sentence ψ, if $\Gamma \vdash \psi$, then $\psi \in \Gamma$. A system is *consistent* if it contains no formula ψ such that both ψ and $\neg\psi$ are theorems.

The structural viewpoint bases the semantic domain on the concept of ***structure*** which in model theory - see e.g. [Bell 1977] - is regarded as an ordered triple

$$\mathsf{U} = \langle A, \{R_i\}_{i\in I}, \{c_j\}_{j\in J}\rangle$$

where A is a nonempty set called the domain of U, $\{R_i\}_{i\in I}$ is the set of all basic $\lambda(i)$-ary relations on A, $\{c_j\}_{j\in J}$ is the set of designated members of A that are constants of U, and I and J are sets indexing the relations and the constants.

An **L-*interpretation*** (or an **L-*structure***) can then be defined as a *structure* U with a domain and a set of constants associated with (corresponding to a perception of) a real world domain **D**, and a mapping which assigns the basic elements of the structure to those of the language **L**. This mapping allows *values, basic operations and basic relations* of U to be assigned to, respectively, the *variable, function and predicate symbols of* **L**. Terms of **L** which do not contain variables denote the constants of U. We shall always use bold type letters to denote language symbols and plain letters to denote structure elements.

An **L-*valuation*** is an **L**-*interpretation* together with a particular assignment of values of its domain to each variable in **L,** so that every *formula* in **L** acquires a true or false truth value.

In terms of these concepts, it results from the previous sections that every *single* dependability claim of a case must be formalised by:

- Three *structures* U_1(environment), U_2(design), U_3 (operational behaviour)
- Three *languages* ($\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$) to express sentences and formulas corresponding, respectively, to statements of system, design and operational subclaims and evidence components.
  A logical system of rules of inference to make formal proofs in these languages.

- $\mathbf{L}_i$- *Interpretations*, i.e. mappings $<\mathsf{U}_i$ - $\mathbf{L}_i>$ which allow values, basic operations and relations of U_i to be assigned to variable, function and predicate symbols of $\mathbf{L}_i$.

If an **L**-*valuation* σ valuates a formula φ to true, we say that this valuation ***satisfies*** φ. If every valuation *satisfying* a set Γ of formulas also *satisfies* a formula φ, we say that φ is a logical consequence of Γ, or that Γ logically implies or ***entails*** φ, and we write "$\Gamma \models \varphi$".

A *sentence* ψ in **L**, i.e. a formula without free occurrence of variable, is logically ***valid*** or ***holds*** in an **L**-*structure* U, or this *structure* U is a ***model*** for ψ, if it valuates ψ to true (we write $\mathsf{U} \models \psi$).
An **L**-*structure* is said to be a *model* for a set of *sentences* Γ if and only if it is a model for every sentence in Γ.

A formal system is ***sound*** if $\Gamma \models \varphi$ whenever $\Gamma \vdash \varphi$. It is ***complete*** if $\Gamma \vdash \varphi$ whenever $\Gamma \models \varphi$. Thus soundness guarantees that every provable formula or sentence is semantically true. Semantic truth is of course essential in the demonstration of safety. An *inconsistent* system cannot be sound. Mathematics and formal systems are sound systems. Thus, whenever we shall require a proof in those systems, the proof implies soundness. Completeness on the other hand ensures that every true fact is provable. Most formal systems of practical interest are not complete. This is why we cannot expect provability whenever we require semantic truth, and not every sound and necessary claim of a dependability claim can be expected to be provable within the same formal system.

8 Entailment and Proof obligations

An $\mathbf{L}_i$-*interpretation* together with its *structure* U_i is the formal connection between the informal statements describing the claims and the evidence associated with the real domain entity i, i=1,2,3, and the formulae of the formal language $\mathbf{L}_i$ used to formulate proofs and rules of inference. In terms of these notions, the claims and evidence components of a dependability case must satisfy to the following obligations:

Every:	**is entailed by:**	**is expanded and provable from:**
Dependability Claim	- - -	environment-system subclaims
Environment-System subclaim	Environment evidence components	Design subclaims
Design subclaim	Design evidence components	Operational-environmentsubclaims
Operational-environment subclaim	Operational evidence components	- - -

Items supported by structure $\cup_1$ and language L_1
Items supported by structure $\cup_2$ and language L_2
Items supported by structure $\cup_3$ and language L_3

***Figure 6*:** Structure and Language Inter-relations

Entailment obligations

To be successful, a dependability case requires that, in each dimension i, i=1,2,3, every subclaim be the *logical consequence* of evidence components in this dimension, i.e. that these evidence components *entail* the subclaim. Besides, the subclaim and the evidence components must reflect actual properties of the real entities.

Therefore, in order to establish this semantic truth, *in each dimension, an L_i*-structure *must exist such that all L_i-valuations* which satisfy the evidence component formulae also satisfy the subclaim formula. If the subclaim is a sentence, *there must exist* an *L_i-structure* which is a *model* for this subclaim.

Proof obligations

Moreover, as we have seen in section 4, in each dimension i, each subclaim is *expanded into* (i.e. tanslated into) subclaims of the next dimension (i + 1). The demonstration that this expansion is "right" - i. e. consistent and complete - requires *the existence of a formal system* in which each subclaim of *dimension i* can be *derived (proven)* from a set of subclaims of *dimension (i+1).*

In other words, semantic truth is required for the use of evidence components to support subclaims, and a formal sound derivation of claims from subclaims is required to establish completeness and consistency of the subclaims. These entailment and proof obligations are summarised in figure 6, and formally expressed in the following sections. It is important to note that this table identifies logical relations only, and does not specify a top to bottom sequence in which subclaims have to be stated, evidence found and proof obligations discharged.

***L*$_1$ - *Interpretations* (environment - system) and obligations**

For a given dependability claim, say a formula or a sentence φ_a, there must exist an $\mathbf{L_1}$-*interpretation*, i.e. an $\mathbf{U_1}$ *structure* which allows the valuation of the sets of $\mathbf{L_1}$- *formulas* and/or *sentences:*

$$\varphi_a \, , \Gamma_{\text{env-evidence}} \text{ and } \Gamma_{\text{es-subclaims}}.$$

where $\Gamma_{\text{env-evidence}}$ and $\Gamma_{\text{es-subclaims}}$ are the expression of the environment evidence components and the environment-system subclaims of φ_a.
At the *semantic* level, *every* $\mathbf{L_1}$-*valuation* which satisfies the set $\Gamma_{\text{env-evidence}}$ must also satisfy the set $\Gamma_{\text{es-subclaims}}$:

$$\Gamma_{\text{env-evidence}} \models \Gamma_{\text{es-subclaims}}$$

In more intuitive terms, there must exist a body of environment-system evidence which, when true in an environment-system structure U_1, makes also true the system subclaim. If both the evidence components and the environment-system subclaims are sentences, it means that U_1 must be a model of both.

At the *syntactic* level, φ_a must be *provable* from $\Gamma_{\text{es-subclaims}}$:

$$\Gamma_{\text{es-subclaims}} \vdash \varphi_a \, .$$

Thus, for *every* dependability claim φ_a, one must have:

$$\forall \varphi_a : \; \exists \, U_1, \mathbf{L_1} \;\; \Rightarrow \;\; (\Gamma_{\text{env-evidence}} \models \Gamma_{\text{es-subclaims}} \vdash \varphi_a \,),$$

and also, because of the soundness of the the formal system: $\Gamma_{\text{es-subclaims}} \models \varphi_a$.

Assume, as an example, that φ_a is the dependability claim of figure 1. Then, the above condition states that there must exist a structure - e.g. equations or a simulation model - corresponding to the thermodynamic behaviour of the coolant in the plant relevant parts (vessel, primary circuit, pressurizer). In this structure, every set of *values* of **T** (temperature) and **L** (coolant level) corresponding to a steam bubble occurrence must valuate to true the condition expressed in the environment-system subclaim of figure 1. Moreover, a language is needed to interpret this structure, to formulate φ_a and the environment-system subclaims from which it must be proved.

L*$_2$ - *Interpretations (design) and obligations

Similarly, for every environment-system subclaim, i.e. for every $\mathbf{L_1}$ formula, say $\varphi_s \in \Gamma_{\text{es\_subclaims}}$, there must exist a design *structure* U_2 and an ***L*$_2$-*interpretation*** which allows the valuation of the ***L*$_2$** formulas and sentences:

φ_s , $\Gamma_{\text{d-subclaims}}$, and $\Gamma_{\text{d-evidence}}$.

where $\Gamma_{\text{d-subclaims}}$, and $\Gamma_{\text{d-evidence}}$ are, respectively, the design subclaims from which φ_s is provable, and design evidence components. At the semantic and syntactic levels, one must have for every environment-system subclaim φ_s:

$$\forall \varphi_s \in \Gamma_{\text{es-subclaims}}: \quad \exists\, \cup_2, \mathbf{L}_2 \quad \Rightarrow \quad (\Gamma_{\text{d-evidence}} \models \Gamma_{\text{d-subclaims}} \vdash \varphi_s),$$

and also, because of the soundness of the formal system: $\Gamma_{\text{d-subclaims}} \models \varphi_s$.

In the case of the monitoring system, assume φ_s is the second environment-system subclaim of figure 1. "Correct executable code" is then a design subclaim, call it φ_d, from which, if true, φ_s must be provable. Evidence to support φ_d may, for instance, be provided by a tool which dissassembles binary code into a higher level language. In this case the individuals and the relations of the structure $\cup_2$ and the $\mathbf{L}_2$-interpretation should be, respectively, the elements of the binary code, the relations of the disassembling process and a $\mathbf{L}_2$ formulation of the disassembled code. The $\mathbf{L}_1$ original formulation of φ_s should be provable from the $\mathbf{L}_2$ disassembled formulation of φ_s. Sections 9 and 10 will discuss the relations between the two pairs, $\cup_1$, $\mathbf{L}_1$ and $\cup_2$, $\mathbf{L}_2$, required by this type of proof obligation.

L_3 - Interpretations (operational behaviour)

For every design subclaim, i.e. for every $\mathbf{L}_2$ formula, say $\varphi_d \in \Gamma_{\text{d-subclaims}}$, there must exist an operational behaviour-environment *structure* $\cup_3$ and an $\mathbf{L}_3$-*interpretation* to allow the valuation of the L_3 formulas and sentences:

$$\varphi_d \text{ , } \Gamma_{\text{o-evidence}} \text{ , and } \Gamma_{\text{oe-subclaims}} \text{ ,}$$

where $\Gamma_{\text{o-evidence}}$, and $\Gamma_{\text{oe-subclaims}}$ are the expressions of the operational evidence and of the operational-environment subclaims from which φ_d. is provable At the semantic and syntactical levels, one must have for every design subclaim:

$$(\forall \varphi_d \in \Gamma_{\text{d-subclaims}}): \quad \exists\, \cup_3, \mathbf{L}_3 \quad \Rightarrow \quad (\Gamma_{\text{o-evidence}} \models \Gamma_{\text{oe-subclaims}} \vdash \varphi_d),$$

and also, because of the soundness of the formal system: $\Gamma_{\text{oe-subclaims}} \models \varphi_d$.

For example, assume that φ_d is the non-functional design subclaim that the monitoring system computations are fail safe in case of invalid input data and cause no spurious alarms (cfr. figure 3). This subclaim is provable from different operational subclaims:

(i) adequate anticipation of post accidental conditions impact on input device and computer equipment
(ii) adequate anticipation of input devices and computer failure modes
(iii) robust (re-)calibration, periodic test and resetting in-service procedures

Evidence for these operational subclaims is provided by:
- operational experience of plant, and accident scenarii,
- operational experience of input device, computer hardware, power supplies and operator behaviour,
- results of integrated tests, probation period, and operators reports.

Individuals of the domain of the operational structure U_3 should be the states of the operator and of the equipments mentioned above. The U_3 relations are determined by accident scenarii, failure modes, operator procedures. Data provided by operational experience and integrated tests must valuate to true the operational subclaims formulated in an $\mathbf{L}_3$-interpretation of this structure.

9 Substructures and Embeddings

Next, in order to understand how the structures U_i and the languages $\mathbf{L}_i$ are inter-related, and the properties that they must satisfy with respect to each other, we shall need the notions of *substructure* and *embedding*.

Let **L'** be a language which is an ***extension*** of **L**, i.e. every **L** symbol is also a **L'** symbol, and, in addition to the predicate symbols and constant symbols of **L**, **L'** contains a set $\{\mathbf{R_i}: i\in I'\}$ of predicate symbols and a set $\{\mathbf{c_j}:j\in J'\}$ of constant symbols.

Given a **L'**-*structure*

$$\mathsf{U}' = \langle A, \{R_i\}_{i\in I\cup I'}, \{c_j\}_{j\in J\cup J'}\rangle,$$

the **L**-*structure*

$$\mathsf{U} = \langle A, \{R_i\}_{i\in I}, \{c_j\}_{j\in J}\rangle$$

is called [Bell 1977] the **L-*reduction*** of U', and U' the **L'-*extension*** of U.

Next, let

$$\mathsf{U} = \langle A, \{R_i\}_{i\in I}, \{c_j\}_{j\in J}\rangle$$
$$\mathsf{U}' = \langle A', \{R'_i\}_{i\in I}, \{c_j'\}_{j\in J}\rangle.$$

be **L**-*structures*. We say that U is a ***substructure*** of U', and we write $\mathsf{U}\subseteq\mathsf{U}'$, if $A\subseteq A'$, for each $j\in J$: $c_j = c_j'$, and, for each $i\in I$, R_i is the restriction of R_i' to A, i.e. $R_i = R_i'\cap A^{\lambda(i)}$, where $A^{\lambda(i)}$ is the set of $\lambda(i)$-tuples of A.

For example, if **L** contains only the predicate symbol = , the function symbols $\{+, \times\}$, and the constants **0** and **1**, then the set of integers is a **L**-*substructure* of the set of rational numbers.

An **L-*embedding*** of U into U' is a one to one mapping T of A *into* A' such that

(i) $\langle c_j, c_j'\rangle \in T$
(ii) $\langle a_1,...,a_{\lambda(i)}\rangle \in R_i \Leftrightarrow \langle a_1',...,a_{\lambda(i)}'\rangle \in R_i'$,

for all $i \in I$, and for $a_1,...,a_{\lambda(i)} \in A$, where $<a_1,...,a_{\lambda(i)}>$ is an ordered sequence of members of A which is an assignment of values to variables $\mathbf{v_1, v_2, v_3}$,... of **L**, and for all pairs a_i, a_i' such that $<a_i, a_i'> \in T$. An isomorphism of $\mathfrak{U}$ onto $\mathfrak{U}$'is an embedding of $\mathfrak{U}$ *onto* $\mathfrak{U}$'.

Stronger relations between interpretations are useful in model theory. $\mathfrak{U}$ is said to be an **L-*elementary substructure*** of $\mathfrak{U}$', and $\mathfrak{U}$' *an* **L-*elementary extension*** of $\mathfrak{U}$ if $\mathfrak{U} \subseteq \mathfrak{U}$' and for any **L**-formula φ all of which free variables are assigned values from the domain of $\mathfrak{U}$, we have:

$$\mathfrak{U} \models \varphi \quad \Leftrightarrow \quad \mathfrak{U}' \models \varphi$$

for all possible value assignments of those free variables, and we write $\mathfrak{U} \prec \mathfrak{U}'$. In other words, a formula (or sentence) φ is *satisfied* (*holds)* in $\mathfrak{U}$, if and only if it is also *satisfied* (*holds)* in $\mathfrak{U}$'.

An **L-*elementary embedding*** is an embedding T of $\mathfrak{U}$ into $\mathfrak{U}$' if for any L-formula φ all of which free variables are among $\mathbf{v_0,...,v_n}$, we have

$$\mathfrak{U} \models \varphi[a_0,...,a_n] \Leftrightarrow \mathfrak{U}' \models \varphi[a_0',...,a_n']$$

for all $a_0,...,a_n \in A$.

It is obvious that if $\mathfrak{U} \prec \mathfrak{U}'$, then $\mathfrak{U} \subseteq \mathfrak{U}'$, but the converse is not true. For instance [Mendelson 1987], if ***L*** has the predicate symbol =, the function symbol + and the individual constant symbol **0**, then the structure $\mathfrak{U}_{2I}$ of even integers is a ***L-substructure*** of the structure $\mathfrak{U}_I$ of integers, but is not an ***L-elementary substructure***: if ϕ**[y] = (∃x)(x+x=y)**, then $\mathfrak{U}_I \models \phi[2]$, but not $\mathfrak{U}_{2I} \models \phi[2]$. Note that $\mathfrak{U}_{2I}$ is embedded in $\mathfrak{U}_I$ with the function g such that g(x)=2x for all x in I. So, an isomorphic substructure is not necessarily an elementary substructure nor an elementary embedding.

Now, let us see how these concepts apply to our dependability case model?

Assume φ_a is a dependability claim sentence, and $\mathfrak{U}_1$ the structure in which φ_a is provable from the set $\Gamma_{\text{es-subclaims}}$ of environment-system subclaims. We have seen in section 8 that an environment-system subclaim, $\varphi_s \in \Gamma_{\text{es-subclaims}}$, must be satisfied in $\mathfrak{U}_1$. φ_s must also be provable in a *sound* formal system from design subclaims, and therefore must also be satisfiable in a structure $\mathfrak{U}_2$ through a $\mathbf{L_2}$-*interpretation*.
Therefore, every $\varphi_s \in \Gamma_{\text{es-subclaims}}$ which must be *satisfied* in $\mathfrak{U}_1$, must also be *satisfied* in $\mathfrak{U}_2$. By the same argument, every $\varphi_d \in \Gamma_{\text{d-subclaims}}$ which must be *satisfied* in $\mathfrak{U}_2$, must also be *satisfied* in $\mathfrak{U}_3$.

For these conditions to be fullfilled, it is obviously *sufficient* that $\mathfrak{U}_1 \prec \mathfrak{U}_2$, and $\mathfrak{U}_2 \prec \mathfrak{U}_3$. Sufficient, but not necessary, because all formulas that must be satisfiable and provable in $\mathfrak{U}_2$ ($\mathfrak{U}_3$) need not be satisfiable or provable in $\mathfrak{U}_1$ ($\mathfrak{U}_2$).

Two propositions on substructures provide tighter conditions. It is possible to demonstrate (see e.g. exercise 2.99 in [Mendelson 1987]) that if $\mathfrak{U} \subseteq \mathfrak{U}'$, then:

(i) Let $\varphi[\mathbf{v_0,...v_n}]$ be a formula of the form $(\forall \mathbf{y_0})...(\forall \mathbf{y_m})\ \psi[\mathbf{v_0,...v_n, y_0,...,y_m}]$ where ψ has no quantifiers. Then, for any $a_0,...,a_n$ in the domain of $\mathfrak{U}$, if $\mathfrak{U}' \models \varphi[a_0,...,a_n]$, then $\mathfrak{U} \models \varphi[a_0,...,a_n]$. In particular, $\mathfrak{U}$ is a model for any *sentence* $(\forall \mathbf{y_0})...(\forall \mathbf{y_m})\ \psi[\mathbf{y_0,...,y_m}]$ where ψ contains no quantifier, if $\mathfrak{U}'$ is a model for this sentence.

(ii) Let $\varphi[\mathbf{v_0,...v_n}]$ be a formula of the form $(\exists \mathbf{y_0})...(\exists \mathbf{y_m})\ \psi[\mathbf{v_0,...v_n, y_0,...,y_m}]$ where ψ has no quantifiers. Then, for any $a_0,...,a_n$ in the domain of $\mathfrak{U}$, if $\mathfrak{U} \models \varphi[a_0,...a_n]$, then $\mathfrak{U}' \models \varphi[a_0,...a_n]$. In particular, $\mathfrak{U}'$ is a model for any *sentence* $(\exists \mathbf{y_0})...(\exists \mathbf{y_m})\ \psi[\mathbf{y_0,...,y_m}]$ where ψ contains no quantifier, if $\mathfrak{U}$ is a model for this *sentence*.

In other words, universal (existential) assertions true in the expansion (in the substructure) are also true in the substructure (in the expansion) if they have no free variables, or if all their free variables receive values from the domain of the substructure [remember that $\mathbf{(\exists y)\psi(y,v)}$ is an abbreviation for $\neg\mathbf{(\forall y)}\neg\mathbf{\psi(y,v)}$].

Corresponding propositions can also be demonstrated for embedded structures.

Translated into our dependability claim model, conditions (i) and (ii) appear to be sufficient for all practical needs. Therefore, it is sufficient that $\mathfrak{U}_1 \subseteq \mathfrak{U}_2$, and $\mathfrak{U}_2 \subseteq \mathfrak{U}_3$, or that these structures be embedded into each other.

Thus, the environment-system structure $\mathfrak{U}_1$ of a dependability claim should be the root *structure* of a tree of structure *extensions*. At the next level of this tree, there should be at most one distinct design *extension* $\mathfrak{U}_2$ per environment subclaim, and one distinct operational *extension* $\mathfrak{U}_3$ per design subclaim at the third operational level. The domain of an *extension* contains the domain its *substructure*. The relations of a *substructure* are the restrictions to its domain of the relations of its *extension*. *Substructures* may be *embedded*, that is related to their *extensions* by means of a relation T between their domains, constants and relations. These T relations express sufficient conditions for maintaining traceability between a dependability claim, its subclaims and their entail and proof obligations.

10 Language Inter-relations

Alike structures, the languages involved in the demonstration that a dependability claim is satisfied or holds in a $\mathbf{L_1}$-elementary substructure $\mathfrak{U}_1$ are not independent.

Let **L'** be a language which is an ***extension*** of **L**, i.e. every **L**-symbol is also a **L'**-symbol. Then for each ***L'****-valuation* σ', there is a unique ***L****-valuation* σ which agrees with σ' on all **L** symbols. We say that σ is the ***L-reduction*** of σ', and σ' is said to be an ***L'-expansion*** of σ. In general an ***L****-valuation* has more than one ***L'****-expansion*.

Let φ be an ***L***-formula.

An ***S-form*** for φ (short for ***satisfiability form***) is a formula **φ'** in some extension ***L'*** of ***L***, such that any ***L****-valuation* σ satisfies φ iff σ has *an* ***L'****-expansion* σ' satisfying **φ'**.
A ***V-form*** for φ (short for ***validity form***) is a formula **φ'** in some extension ***L'*** of ***L***, such that any ***L****-valuation* σ satisfies φ iff *every* ***L'****- expansion* σ' of σ satisfies **φ'**.

It results from section 8 that *the language* $\mathbf{L_2}$ *should be an extension of* $\mathbf{L_1}$. More precisely, for every dependability claim φ_a, the language $\mathbf{L}_2$, in which are expressed the design subclaims $\Gamma_{\text{d-subclaims}}$ from which an environment-system subclaim φ_s is provable, should be an "extension" of the language $\mathbf{L}_1$ in which φ_s is expressed. Moreover, the formula φ_s should have a *satifiability expansion* in$\mathbf{L_2}$.

Similarly, $\mathbf{L}_3$ in which are expressed the operational subclaims from which a design subclaim φ_d is provable, should be an *extension* of $\mathbf{L}_2$ in which this design subclaim φ_d is expressed.
Moreover, the formula φ_d should have a *satifiability expansion* in $\mathbf{L_3}$. Clearly, satifiabilityforms are necessary and sufficient to guarantee that subclaims remain satisfied in their expanded structures.

These relations among structures and languages across subclaim dimensions are indicated by shaded areas in figure 6.

11 Validation and Design

Ideally, the techniques used in practice for the design and the validation of dependable computer systems should conform to the formal notions introduced in the previous sections. However, for real systems of reasonable complexity, the demonstration that the implementation satisfies the dependabilty claims, especially their non-functional requirements, is quite hard. *Design criteria*, and *requirement refinement* are some of the classic concepts used to circument the difficulty. More recently, object oriented techniques like *reflection* and *meta-object* protocols have also been proposed (see [Fabre 1998]). The relationship between these concepts and the formal model outlined in the previous sections is briefly discussed below.

- *Design Criteria*

The *real domain* (**D)** designers have to start from in practice, consists of informal statements on: (i) the environmental properties and constraints that exist and must be maintained by the system; (ii) the application dependability requirements (probabilistic or deterministic), and (iii) evidence provided by the environment and the application (in terms of possible pre-existing safe states, lines of defense, redundancy or diversity, operational procedures at the application level).

From these elements, designers usually *infer* appropriate *design criteria* (see e.g. [Börger 1997]), in certain industrial sectors called design assumptions or safety design principles. If satisfied by the real system implementation, the selected criteria are assumed to somehow guarantee the prescribed dependability requirements, especially the non-functional ones. Instances of design criteria are: the single failure criterion, stable interfaces in time or value domains [Kopetz 1998], fail silent behaviour, isolation between safety and non-safety functions, safety margins. In practice, the selected criteria are used to translate non functional requirements of the application into requirements for the design and the implementation.

In many fields of application, the adopted design criteria result from a general engineering consensus and are imposed by standards. They are sometimes taken - rather arbitrarily - as deterministic substitutes for non-functional system requirements. Design criteria, however, are and should be considered as proper environment-system subclaims (as in the example of figure 3) from which the satisfaction of dependability claims needs to be demonstrated. An environment-system structure or model, and a language interpretation are needed for this proof obligation which, contrary to practice, should be considered as an essential part of the dependability case. Formally, the valuation of this structure should show that the design criteria are the logical consequence of environment-system evidence. The language interpretation should allow the dependability claim to be proven from the design criteria.

- *Refinement Process*

In practice, because of the complexity of the design and its implementation, the demonstration that the design criteria are satisfied by the implementation should be - like the design itself - carried out in several steps. A first specification of the design is shown to be a *model* for the selected design criteria. This specification can then be refined into an implementation through successive reification stages. The concepts of *substructure* and *embedding* can be used to formalise the syntactic and semantic aspects of these refinement stages, and also the conditions under which they preserve the design criteria. At the syntactic level, the successive language expansions used at each stage should then be shown to preserve *satisfiability forms* of the initial design specification and of the design evidence.

- *Reflection and Metaobjects*

The use of object-oriented techniques such as reflection and meta-object protocols has recently been proposed for implementing non functional requirements and fault tolerant mechanisms into a dependable application (see, e.g., [Fabre 1998]). These generic mechanisms have their advantages but also their limits due, in particular, to the difficulty of the validation of their behaviour within a particular application (see e.g. [DeLemos, Romanovsky, 1997]). This debate is outside the scope of this paper; but it is interesting to note that the formal model of section 8 may provide a formal description of these concepts.

As an example, consider the non-functional dependability claim of figure 3. The different attributes a PAM (post accident monitoring) function must enjoy - which are listed as design criteria in figure 3 - may be associated to a class of metaobjects. At the semantic level, the

valuation of an environment-system structure, say U_1, by environment evidence components must establish that an application object (function) specification together with these metaobjects specifications will behave as intended by the dependability claim. At the syntactic level, this dependability claim should be provable through a language interpretation from the object and meta-object external specifications. The structure U_1 must be expanded into the design structure U_2 of the functional object and of the meta objects. In this design structure, design evidence must valuate to true the design specifications from which the external specifications of the functional object and the metaobject must be proven. The *embedding* $T_{1,2}$ of the elementary structure U_1 into a structure U_2 is a relation which is a formal representation of the metaobject design reflection mechanism. The domain of U_1 is the *domain* of the reflection mechanism $T_{1,2}$, the domain of U_2 is the *range* of the reflection.

Similarly, a design structure U_2 could be extended into an operational structure U_3 of the run-time behaviour of the functional object and of the metaobjects. The embedding $T_{2,3}$ of U_2 into U_3 formally corresponds to a run-time metaobject reflection mechanism.

An issue is whether each of these structures U_1, U_2, U_3 can be constructed as a set of separate substructures, one associated with a functional object, and one associated with each metaobject applied to the functional object. If not, the metaobject protocol does not enjoy the semantic property of composability which is essential to the usefulness of the concept. The conditions that substructures must satisfy for composability could presumably be explored with model theory. The properties required from metaobject and reflection mechanisms in order to be validatable might also be identifiable. This work is in progress but preliminary results already raise interesting questions.

Suppose, for instance, that U° is a structure corresponding to some functional object:

$$U^\circ = <\{A, E, B\}, \{R_i^a\}_{i\in I^a} \{R_i^b\}_{i\in I^b}, \{c_j\}_{j\in J}>$$

where the elements of A are the non-faulty states of the object, E is a set of fault conditions, B is a set of faulty states, $\{R_i^a\}$ is the set of relations corresponding to transitions between states of A, and $\{R_i^b\}$the set of relations corresponding to transitions from $\{A\cap E\}$ to B, and $\{c_j\}$ a set of constants.

We would like to construct a meta-object which, when applied to (reflected upon) a functional object transforms this object into a fault tolerant object which, instead of moving to states of B when faults are activated, moves to a set of safe states, say S. The fault tolerant ***L****-environment-system structure* which should result from the application of the meta-object on U° should be:

$$U_1 = <\{A,E,S\}, \{R_i^a, R_k^s\}_{i\in I^a\ k\in I^s}, \{c_k\}_{k\in K}>$$

where $\{R_k^s\}$is the set of transitions from $\{A\cap E\}$ to S and has the same domain as $\{R_i^b\}$, and where $\{c_k\}_{k\in K}$ contains $\{c_j\}_{j\in J}$.

The meta-object structure, say U^{mo}, should be applicable independently from the specific functionality of U°, essentially defined by A and $\{R_i^a\}_{i\in I^a}$. It should therefore be restricted to:

$$U^{mo} = <\{E,S\}, \{R_k^{es}\}_{k\in I^{es}}, \{c_q\}_{q\in Q}>$$

where the domain and the range of $\{R_k^{es}\}$ are E and S, respectively, and $\{c_q\}_{q\in Q}$ is a subset of $\{c_k\}_{k\in K}$.

First, we observe that the structure U^{mo} is a substructure of U_1 *iff* R_k^{es} is the restriction of R_k^{s} to E. In other words, the transitions to B or S must depend on E only, and the safe states in the fault tolerant metaobject must be independent from the current operational state in A. This may be a serious restriction in practice.

A second observation is that $U°$ is neither a *substructure* nor an *L-embedding* of U_1. The domain of $U°$ is not contained in U_1, and the relations of U_1 cannot be restricted to those of $U°$. Therefore, formulas or sentences which are satisfied in $U°$ are not necessarily satisfied in U_1 and conversely.

A third observation is that U_1 is not the result of a *direct product* construction of $U°$ by U^{mo}. The direct product is defined as followed for a simple case (see e.g. [Bell 1977]). Let for each $i\in I$, where I is an arbitrary non empty index set, $U_i = <A_i,R_i>$ be an **L**-*structure* with a non-empty domain A_i and a single binary relation R_i. Let $\Pi_{i\in I}\ A_i = A$ be the cartesian product of the sets A_i. Let f,g denote elements of A. The direct product $\Pi_{i\in I}\ U_i$ of the family $\{U_i : i\in I\}$ is the structure $<\Pi_{i\in I}\ A_i,Q>$, where Q is the set of all pairs $<f,g>$ such that $<f(i),g(i)>\in R_i$. It is clear that, in the case of the metaobject, the domain of U_1 is not the cartesian product of that of $U°$ by U^{mo}.

Conditions for the composability of structures remain to be explored. With the use of ultrafilters, fundamental results in model theory like the Los' theorem on ultraproducts (see e.g; [Bell 1977, Mendelson 1987]) might provide a way to identify the conditions under which given formulas which are satisfied in $U°$ and U^{mo} are also satisfied in U_1.

12 Open Issues and Concluding Remarks.

This work is only a preliminary step towards the identification of structures, interpretations and languages necessary for the semantic, syntactic and logical treatment of an application dependability claim, and for the analysis of the convergence of its associated evidence components. Further work is needed to explore the properties of these structures; but the mere fact that this first step was feasible is encouraging in itself. Models of the kind we have discussed should contribute to more reliable and efficient dependability cases, in particular by helping licensors, designers and licensees to better understand each other.

On the other hand, the proof obligations and the language properties required by the theory appear to be quite constraining, and probably difficult to realise in practice. These obligations and properties should therefore be essentially regarded as guidance for reforming and improving design and validation practices, at least for systems in need of a strict validation and licensing process.

The simple axiomatic model outlined here offers of course various possibilities for enrichement and further investigation.

First, the model implicitly shows that *each* dependability claim of an application can be expanded into subclaims and dealt with separately from other claims on the same application. This separation of concerns contributes to traceability, and - although not in current usage - should be recommended in practice. It also allows for the "reuse" of subclaims and evidence components for different claims, and in principle allows systems to be composed of subsystems "imported" with claims transposed as subclaims in the dependability case of the composed system.

Secondly, we have dealt with deterministic and positive evidence only, but the approach could be extended to integrate plausibility or probabilistic measures of confidence associated with components of evidence, and possibly components of counter-evidence. At any rate, the *topology* - nodes and edges - of a belief bayesian net for an application claim (see e.g. [Courtois 1998]) - instead of being arbitrarily or subjectively derived - could be more formally based on a three dimensional structure.

Third, the separate treatement of claims, however, leaves aside the issue of demonstrating the completeness and consistency of the *set* of claims which compose a dependability case. Another dimension added to the model - i.e. an application based structure and interpretation like a complete and consistent set of postulated initiating events (PIE's)- is a possible way to integrate this demonstration.

Finally, for the simplicity of the analysis, we have considered the dependability case as a static, complete case, all evidence components being available. In reality, of course, the case evolves as the project progresses. In the earlier phases, environment and system evidence only is available. Then design evidence accumulates as the hardware and software are being developed. Once (parts of) the system can execute, operational evidence of actual experience and maintainability becomes available. The structure of a dependability case, as shown in figure 6, with its successive subclaim expansions and proof obligations is suitable to allow this progression.

Acknowledgements

This work was supported by the European Community Esprit Long Term Research Project 20072-DeVa (Design for Validation). Thanks are due to colleagues in this project, Friedrich Von Henke (Ulm University), Bev Littlewood and Lorenzo Strigini (City University, London), Marie-Claude Gaudel (LRI, Paris), Jean-Claude Laprie (LAAS, Toulouse), and Peter Puschner (Vienna University), for their useful comments on the formal and dependabilty aspects of this research.
The author is also very much indebted to his colleagues in AV Nuclear, especially Ray Ashley, Claude Baudelet, Pierre Govaerts, Jean-Jacques Van Binnebeek and Jacques Vlassenbroeck, for their advice and/or the sharing of their experience in nuclear instrumentation, operations and safety. Thanks are also due to Michel Sintzoff, department of computer science, University of Louvain-la-Neuve, for his comments. Last but not least, this research work has been initially motivated and greatly influenced by D.L. Parnas, Mc

Master University, Canada, more specifically by his work on documentation, and by his comments on an earlier version.

References

[Bell 1977] J. Bell, M. Machover, "*A Course in Mathematical Logic*", North Holland, 1977, 2nd printing 1986.

[Bloomfield 1995] R. Bloomfield, "The SHIP Safety Case", in *Safecomp 95, Proc. 14th IFAC conf. on Computer Safety, Reliability and Security* (G. Rabe ed.), Belgirate, Italy, 11-13 October 1995, Springer, ISBN 3-540-19962-4.

[Börger 1997] E. Börger, L. Mearelli, "Integrating ASM's into the software Development Life Cycle", Manuscript presented at the Schloss Dagstuhl International Seminar on "*Practical Methods for Coe Documentation and Inspection*", Saarbrücken, May 1997.

[Courtois 1998] P.-J. Courtois, N.E. Fenton, B. Littlewood, M.Neil, L. Strigini,, D.R. Wright, "Bayesian Belief Network Model for the Safety Assessment of Nuclear Computer-Based Systems." Esprit Long Term Research Project 20072-DeVa, Second Year Report Part 2, pp.485-512.

[Dahll 1997] G. Dahll, "Safety Assessment of Software based Systems". SAFECOMP'97, York 1997.

[DeLemos, Romanovsky, 1997] R. Delemos and A. Romanovsky, "Coordinated Atomic Actions in Modelling Objects Cooperation", in 1st IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing, Kyoto, Japan, April 1998. (technical report, dept. of Computer Science, Univ. of Newcastle upon Tyne, TR 620, 1997).

[Fabre 1998] J.-C. Fabre, T. Pérennou, "A Metaobject Architecture for Fault Tolerant Distributed Systems: The FRIENDS Approach." Esprit Long Term Research Project 20072-DeVa, Second Year Report Part 2, pp.195-228.

[Kopetz 1998] H. Kopetz, "Component-Based Design of Large Distributed Real-Time Systems". Esprit Long Term Research Project 20072-DeVa, Second Year Report Part 2, pp.17-30, 1998.

[Mendelson 1987] E. Mendelson, "*Introduction to Mathematical Logic"*, Wadsworth & Brooks/Cole, Monterey, third edition, 1987.

[NUREG 0696] " Functional Criteria for Emergency Response Facilities" NUREG - 0696 Report. Division of Emergency Preparedness, Office of Inspections & Enforcement, U.S. Nuclear Regulatory Commission, Washington, D.C. 20555, 1981.

[Parnas et al. 1991] D.L. Parnas, G.J.K Asmis, J. Madey, "Assessment of Safety Critical Software in Nuclear Power Plants" Nuclear safety, 32, 2, 1991.

[Rushby 1993] J. Rushby, "*Formal Methods and the Certification of Critical Systems*", Technical report CSL-93-7, SRI International, CA, December 1993.

[Wong 1998] Ken Wong. "Looking at Code With Your Safety Goggles On." *Proc. 1998 Ada-Europe International Conference on Reliable Software Technologies*. Lecture Notes in Computer Science 1411, Springer, pp.251-262.