(In) Measurement Journal, Vol 10, No 2, Apr-Jun 1992, pp.79-86.

Using Software Specification Methods for Measurement Instrument Systems

Part 1: Structured Methods

L. Finkelstein(*), J. Huang(*), A.C.W. Finkelstein(+) and B. Nuseibeh(+)

(*) School of Engineering, City University, London EC1V 0HB, UK
(+) Department of Computing, Imperial College, 180 Queen's Gate,
London SW7 2BZ, UK

Investigation has been undertaken into the use of software specification methods for instrument systems specification. In the first part of this paper, some of the well-known structured software specification methods are briefly evaluated against a set of criteria we established, in the context of measuring instrument systems. We then conduct a case study in the widely used CORE method. It has been established that CORE can provide systematic and effective support for functional requirement analysis of complex instruments and instrument systems. Its decomposition-based analysis helps to create complete and consistent functional requirement specification.

Keywords: Requirements specification, Measuring instrument systems, Structured methods

1. Introduction

The process of requirements specification is recognized as a critical activity in the development of engineering systems. Validated requirements specification documents may be used to legally bind clients and their suppliers, and form a vital reference against which designs may be verified.

Requirements specification techniques have developed separately in the instrument and software engineering communities, and despite advances in both camps, little effort has been made to study the cross-fertilization of these developments across the communities' boundaries. As part of the SEED (Software Engineering and Engineering Design) project, an investigation into transfer of software specification methods to instrument systems' specification is underway [Finkelstein et al, 1990a]. Furthermore, many modern instrument systems typically comprise of a software component or subsystem, making such technology transfer studies all the more relevant.

2. Requirements specification of instrument and software systems

Let us start by providing a unified definition of requirements specification. We view a requirements specification as a declarative description of the services a system should provide, the constraints under which it must operate and standards the delivered system must meet. A service may be an input-output transformation, materializing the inputs and outputs into data in software systems, or power, signal or data in instrument systems. Constraints typically include physical restrictions on the system such as size, weight and operating/storage conditions. Standards the delivered system must conform to may include reliability and maintainability.

The specification of services is known as the functional specification, whereas the specification of constraints and standards as non-functional specification. The functional aspects of a specification must describe a system's interface and its performance. Interface specification includes the source/destination, type, range and format of inputs/outputs, whereas performance specification may include dynamic response characteristics, system response to external disturbances and exception conditions such as failures of power, hardware, software, and so on. Non-functional specification encompass descriptions of system reliability, limits on memory size, levels of operating/storage/transportation temperature, pressure and humidity, physical characteristics such as size and weight, maintenance specification, and so on.

With the notable exception of the defense industry [MIL-STD-490A, 1985], most requirement specification documents in the instrument industry are produced without the explicit use of specification methods. Frequently these documents are generated by the modification of specifications of similar systems. On the other hand, with the "software crisis" as a driving force, the software engineering community has been rapidly developing a large number of specification methods. These have alleviated many of the difficulties of software development, and their applications to the instrument community appears an attractive proposition. What follows is a brief investigative review of some of the popular software specification methods and their applicability to instrument system specification.

3. Requirements Specification Method

Requirements specification methods provide prescriptive step-by-step guidance for the production of requirements. They describe how and in what circumstances certain actions may be performed during the specification process. Specifications may be produced by the same method using different notations.

Predominant in the software engineering community are two types of specification methods: formal and structured informal methods. In a formal method, the vocabulary, syntax and semantics used for constructing a specification document are explicitly defined [Cohen, 1986]. Formal specifications are therefore based on mathematical foundations, including set theory and predicate logic. Although not based on mathematics, structured methods are intended to bring a level of modularity and structure to specification documents and their development process. Their main attraction lies in their explicit representation of the structure of the functional aspects of specifications. Some of these methods are based on system modelling, including data decomposition. Here details of system inputs, outputs and actions (i.e. data processing activities) are identified. In the remainder of this paper, we shall direct our efforts towards evaluating the applicability of such structured methods for the specification of instrument systems. Results of study in formal specification methods will be presented in the second part of the paper.

4. Evaluation Criteria for Requirements Specifications

A number of authors have set down software specification evaluation criteria [Cohen, 1986; Birrell & Ould, 1985; Balzer, 1986]. Although there is a variation in emphasis, these criteria equally apply to the requirements specification of instrument systems. They include understandability, testability, consistency, completeness, relevance, correctness, feasibility and maintainability of the specifications.

The understandability criterion of such a specification requires it to be clear, unambiguous and concise. When a specification is a contract between a client and the supplier, all efforts should be made to avoid misunderstandings between the two sides. In particular, a specification should not have ambiguous statements; that is, there should be

only one way to interpret the specification.

A specification should also be testable. Unquantifiable requirements such as 'as much as possible' should be avoided.

Consistency requires that no two requirement statements contradict each other, and words and terms have the same meaning throughout the document. Thus, mutually exclusive statements and clashes in nomenclature should be avoided.

Completeness, relevance and correctness emphasize the need for a specification to satisfy the client's requirements. Completeness implies that all client's needs are met, relevance requires that only the client's needs are met, and correctness requires that the client's needs are correctly expressed. Incomplete, irrelevant or incorrect specifications may cause a considerable amount of legal and/or investment cost to all parties concerned.

A specification is feasible if and only if it can actually be satisfied using the available resources, including tools, techniques, people and investment. This may mean carrying out high-level designs, prototyping and third party consultancy.

Maintainability is important because specifications typically change and develop over time. Specifications should thus be easy to modify, to minimize the costs for both client and specifier.

Using the above criteria, we are now ready to evaluate a number of software specification methods for the specification of instrument systems.

5. Requirements Specification Methods for Software Systems

This section of the paper presents a brief introduction to some of the well-known requirements specification methods adopted in the software industry. We shall discuss two categories of such methods, namely, those based on data-flow diagrams, and those on entity-relationship diagrams. Data flow diagrams (DFDs) define the data flows between

various functional processes. A process is a transformation of input data into output data according to some function. While a DFD diagram shows details of data transaction processes, the purpose of an entity-relationship diagram (ERD) is to show the data entities themselves and how they are related to one another. DFDs include both processes and data, whereas ERDs concentrate on the data entities only. Entities are usually objects, for example, customers, suppliers, parts, products. A relationship may exist between two objects, e.g., a supplier "offers" a product, and a product "is bought from" a supplier.

In Structured Analysis (SA) [De Marco, 1978], a hierarchy of DFDs is first produced to define the data flow between various functional processes and data stores. These hierarchies are essentially top-down decompositions of the transformations of system inputs to outputs. Named vectors, bubbles, straight double lines, and boxes are used to represent data flows, processes, data stores and mere sources/sinks of data, respectively. A Data Dictionary (DD) then provides a top-down decomposition of the data that appears in the DFDs. DD definitions are given in terms of relational operators such as 'sequence', 'selection', and 'iteration'. For example, 'coupled_signal' can be one of 'AC coupled signal', 'DC coupled signal', 'LF coupled signal' and 'HF coupled signal' (selection). Finally, the processes in the lowest-level DFDs are called Primitive Processes, and they are defined in so-called mini-specs using other specification tools, e.g. Decision Tables.

COntrolled Requirements Expression (CORE) [Systems Designers, 1985] is another popular requirements specification method. A fundamental concept of CORE is that of *viewpoints*. A viewpoint can be a physical or functional entity which stores or processes information. Functions, subparts and components of a system, together with its environment and operator can all be modelled as different viewpoints. In CORE, the analysis and specification of user requirements is performed in a step-by-step manner through a number of well-defined stages: Problem Definition, Viewpoint Structuring, Tabular Collection, Data Structuring, Single Viewpoint Modelling, Combined Viewpoint Modelling and Constraints Analysis. Problem Definition, together with Constraints Analysis, describe a system's non-functional requirements, such as business objectives, cost, timescale, reliability and environmental constraints. A system is then modelled as a hierarchy of viewpoints at the Viewpoint Structuring stage. The various levels of viewpoints are then modelled via input-action-output forms during Tabular Collection and

Single Viewpoint Modelling - the latter introducing additional timing and control information. Input and output data associated with the viewpoints are then decomposed during the Data Structuring stage. Finally, the Combined Viewpoint Modelling stage produces an overall representation of the key processing activities across the entire system. CORE can be used to analyse general information processing systems, be they software, mechanical, electronic or social in nature.

Software Requirements Engineering Methodology (SREM) [Alford, 1980] models a system as paths of data-processing in the from of R-NETs. These paths consist of input messages, a sequence of processing tasks involving both data control flow, and output messages. Paths may branch out under different control conditions. R-NETs can be represented in a machine-readable Requirements Statement Language (RSL) which is then translated into abstract form in the REVS (Requirement Engineering and Validation System) relational database. Specifications in the REVS database may then be formally checked for consistency and completeness. Furthermore, REVS may also be used in a Pascal-coded simulation session to check whether algorithms exist to meet the specifications. Although parts of SREM can be used manually, the SREM/REVS combination offers more benefits, such as automated completeness and consistency checking, and automated requirements documentation.

A popular method used in software industry is Jackson System Development (JSD) [Jackson, 1983]. Its six steps extend across the entire software development phase, including specification, design and implementation. A JSD specification is not based on top-down functional decomposition and does not incorporate the concept of data flow. Instead, it models the real world in terms of entities, and provides guidelines for the analysis and structuring of these entities. In JSD, the distinction between specification and design is not immediately apparent, as the first four steps all involve the user, and design starts at the second step. Nevertheless, specification in JSD is largely carried out during the first two steps, i.e. entity action and entity structure steps. Relevant real world entities and their actions are listed during the first step. The lists may be as comprehensive as possible to start with, but a number of guidelines are then applied to reduce the lists. During the second step, entity structure diagrams for each entity listed during the first step are produced. Such a diagram shows the actions of the entity in terms of sequence, iteration

and selection components. In other words, actions can be iterative, which are repeated zero or more times; selective, only one of which takes place; or sequential, which are ordered in time. Therefore, the diagram represents the time-ordering actions that constitute the lifetime of the entity. Although entity action lists and entity structure diagrams do not form a functional specification, they may greatly promote understanding between a system analyst and the client.

Other specification techniques are being researched and developed. Some take the more traditional approaches of information modelling such as DFDs and ERDs, and attempt to develop formal specifications from these (e.g. Structured Common Sense, or SCS, developed at Imperial College [Finkelstein and Potts, 1987], which guides the analyst towards producing formal specifications in Modal Action Logic). Other methods take a more "revolutionary" approach, exemplified in Object Oriented Analysis (Coad and Yourdon, 1990).

6. Assessment of Current Methods for Instrument Systems Specification

We use the following criteria to evaluate the suitability of a particular method for the requirements specification of instrument systems:

1. Does the method help and guide the specifier to produce an understandable, consistent, complete, relevant, correct and feasible specification
2. Is the system model adopted in the specification method based on input-output transformations ? [Finkelstein & Finkelstein,1987]
3. Does the method have supporting CASE tools ?
4. Does it provides guidelines for validation; e.g. for checking completeness and consistency ?

CORE provides explicit rules for completeness and consistency checking for the functional aspects of specifications. Unlike most of the methods described, CORE also provides guidelines for eliciting non-functional requirements - although these are not expressed in any formal notation. Supporting CASE tools are available from Systems Designers and British Aerospace, and experimental prototypes have been developed at

Imperial College [Nuseibeh, 1989].

SREM and SCS may also be feasibly applied to instrument system specification. SREM has powerful supporting tools in its software support environment REVS, while the temporal analysis of SCS is a useful feature frequently ignored by the other techniques.

7. Specifying an instrument system using CORE

To illustrate the advantages and problems associated with using CORE for instrument system specification, we now conduct a case study of a simplified cathode-ray oscilloscope (CRO).

The first stage of specification using CORE is Problem Definition, in which the Customer Authority, Business Objectives, Principle Features of the System, Costs and Timescales, and so on, are identified. We assume that the customer has given the following functional description of a simplified single-channel CRO. It accepts a voltage signal as input, and displays a plot of the input signal against time, on the scope's screen. Further analysis is needed to produce a specification document, which will have to be approved by the customer. While customer requirements must be satisfied in the final product, the development process of this oscilloscope will have to be subject to time and financial constraints. The production of such a detailed problem definition report involves managerial, technical, financial and marketing personnel. Therefore, it will not be pursued further in this paper. However, this phase of system analysis has been well-documented in engineering system analysis literature [Sage, 1977].

Having identified the problem, a functional structure of the system should now be constructed in the "Viewpoint Structuring" diagram (Figure 1). A *viewpoint* can be a physical or functional entity of the system under analysis. The term viewpoint is referred to here as a functional entity of the oscilloscope system. First of all, "oscilloscope" can be chosen as the top-level viewpoint of such a system, and "CR display" as a lower-level viewpoint. The problem now is to construct other lower-level viewpoints such that they form part of the input-output transformation function.

We start from the output end of the transformation process. Two sets of plates, supplied with voltages, are used to deflect a beam of electrons, producing a light spot on the CR display screen at a certain position (X,Y). The vertical deflection plates are driven by the input voltage signal, whereas the horizontal plates by a ramp voltage, thus causing the light spot to sweep from left to right on the screen at a constant speed. Therefore, we have another two lower-level viewpoints under "oscilloscope", i.e. "vertical deflection system" and "horizontal deflection system".

The input signal may have an AC component with a big DC bias, and it is usually desirable to see only the AC component on the oscilloscope. Therefore, an "Input Coupling Unit" can be used to select the right coupling option. Since the input voltage is usually too weak to cause the vertical deflection, we need an amplifier for the input signal. Consequently, a viewpoint "Input Amplifier" is added under "vertical deflection system". Usually, an "Input Attenuator" is also needed to produce user-adjustable sensitivity range (mV/cm or V/cm).

Under "Horizontal deflection system", we need a "Ramp generator" which accepts a series of impulse signals and produces a corresponding series of ramp voltage signals to drive the horizontal plates. The impulse signals are generated by a "Trigger" on receiving what is called a "trigger signal". Such a trigger signal can be the (amplified) input signal itself, an external trigger signal, or a (50Hz) line signal. The operator selects an appropriate trigger signal through a "Trigger mode unit", which may provide AC, DC, HF, or LF coupling for the external trigger signal or the input signal. However, a coupled signal can not directly be used by the trigger until the triggering polarity and triggering threshold level are determined by the operator through, say, a "Triggering Polarity and Lever Selector".

Having obtained the various levels of viewpoints, i.e. functional units of the oscilloscope, we are now ready to draw the "Viewpoint Structuring" diagram, as shown in Figure 1. This diagram will have to be approved by the Customer Authority.

It can be seen that the environment is modelled as an "indirect" viewpoint, i.e. a sole source or destination of signal, instead of a "direct" signal processing viewpoint. Note that

signal flow is analysed throughout this study, rather than data flow. The operator only provides control to the system, e.g. the selection of trigger mode.

Tabular Collection Forms for each of the viewpoints are then produced, explicitly stating the signal processing activities (Actions) of the direct viewpoints, the type and source of input signals and the type and destination of output signals. Completeness requires the analysis of all the relevant actions, signal inputs and signal outputs of all direct viewpoints. All signal inputs must have a source and all signal outputs must have a destination. Consistency also needs to be checked; for example, an output flow into a direct destination viewpoint in one table, must be the same input signal in another table for this viewpoint. There are 9 direct viewpoints in this study, and therefore 9 tabular collection forms must be produced. Two of these forms are given in Figure 2 for demonstration purposes.

Next, at the Data Structuring stage (Signal Structuring to be precise), the signal outputs of each direct viewpoint are analysed, giving top-down decomposition diagrams similar to the viewpoint structuring diagram. Again, there are 9 such diagrams. Figure 3 shows that the output of Trigger_Coupling_Unit, i.e. Coupled_signal, can be one of AC_coupled_input_signal, DC_coupled_input_signal, LF_coupled_input_signal, HF_coupled_input_signal, AC_coupled_external_signal, DC_coupled_external_signal, LF_coupled_external_signal, HF_coupled_external_signal, and Line_signal.

Single Viewpoint Modelling is then performed for each Viewpoint, resulting in essentially another diagrammatic interpretation of the information in the Tabular Collection Forms. The extra information at this stage includes inter-action data flows, the control of actions (i.e. these actions are not primarily caused by the flow of input data, but by an external control from another viewpoint, e.g. the "Operator"), and time-ordering of actions. The Single Viewpoint Model diagram for the Trigger_Coupling_Unit is given in Figure 4. The symbol $\mathbb{I}$ represents the so-called "channel flow" signal, i.e., all values of which must be processed.

The final diagrammatic representation is known as the Combined Viewpoint Model diagram, in which signal processing that occurs between viewpoints of the entire system is

represented. This diagram gives a systematic view of the signal flows and processing actions across the whole system. The Combined Viewpoint Model diagram in Figure 5 shows one possible system scenario, in this case based on all the Single Viewpoint Model diagrams.

Having completed functional analysis of the oscilloscope, from Viewpoint Structuring to Combined Viewpoint Modelling, the functional aspects of a specification document can be abstracted out from the Combined Viewpoint diagram, identifying data-flow and control-flow interfaces between the system and external indirect viewpoints. These aspects of functional behaviour are presented in Figure 6. As can be seen, this diagram gives a more detailed insight into the functionality of the oscilloscope then the initial problem statement from the Customer Authority. This description, of course, must be approved by the Customer Authority. A full functional specification for the oscilloscope, which defines details for input, output and control, can be given as follows (Numbers of the statements correspond with the numbers in Figure 6 for convenience of understanding).

General description: The function of the oscilloscope is to display traces of voltage signal against time.

(1) Input signal: The bandwidth of input signal can range from 0 to 20MHz. The rise time of a step response should be less than 20ns, with an overshoot of no more than 1%. Input impedance must be 1MΩ || 30pF. The maximum value of input signal must not exceed 400V (DC + peak AC).

(2) External trigger signal: The bandwidth of external signal should be from 0-40MHz.

(4) Input coupling option: The operator selects one of three input coupling options, i.e. AC, DC or GND (ground).

(5) Sensitivity range: (Vertical deflection) sensitivity should be variable between 5mV/cm and 20V/cm in 12 steps.

(6) Triggering mode and coupling option: Triggering sources include the input signal itself, an external signal, or the 50Hz-line signal. The operator selects triggering mode (input, external or line signal) and coupling option (AC/DC/HF/LF). The coupling options only apply to

external signal and input signal. HF and LF options should allow signal components of 100kHz (-3dB) - 40 MHz (-3dB), 0-100kHz (-3dB) to pass through, respectively.

(7) Triggering polarity and level: Triggering can be from positive or negative slope, with threshold level of 5mm on the scope for input signal, or 0.6V for external signal.

(8) Timebase: Timebase should be adjustable from 0.5us/cm to 0.2 s/cm in 18 steps.

(9) Traces: The CR screen should be rectangular, with internal graticule of 8 x 10 cm.

The last stage in CORE is Constraints Analysis. Virtually all the requirements which can not be analysed in the diagrammatic sections should be described at this stage. They include non-functional requirements such as physical constraints, cost and timescale constraints, power consumption, and so on. Non-functional requirements can usually be produced using traditional checklisting techniques, which may vary from company to company. The following gives a (non-exhaustive) list of non-functional statements:

(10) Physical characteristics: Weight 7-8kg. Width 280-300mm. Height 140-150mm. Depth 380-390mm. Colour blue.

(11) Power consumption: 35-40W.

(12) Cost: No more than £350.

(13) Timescale: The development process must not take more than 8 months, including design and manufacturing.

The end product of a CORE analysis should be an actual specification document, which may consist of textual statements, tables, diagrams and, if necessary, some mathematical expressions. Unfortunately, there exists no standard for the specific format of this document in the instrument community. Different companies tend to create and adopt their own specification style and format. Therefore, apart from the functional and non-functional statements as given above, the authors do not attempt to produce a complete specification document in this particular study.

The advantages of using CORE for the requirement analysis of this simplified oscilloscope can now be discussed. A step-by-step analysis of information flows and information processing activities can be performed using the CORE notation, with each preceding step providing the basis for the current step of specification. This analysis helps gain insight into the functional behaviour of the system under study. Also, completeness and consistency checking on information transactions can be performed. Therefore, the specific benefits of using CORE for functional requirements analysis are its explicit graphical notation, and completeness and consistency checking rules. These benefits can be best appreciated in complex instrument systems.

However, CORE loses much of its attraction if a measuring instrument or instrument system (e.g. a pressure gauge), does not have many signal processing functions. A CORE analysis document for such applications would then have a much larger Constraints Analysis section than all the other sections put together. Obviously this problem is not unique to CORE - there are no accepted methods for expressing non-functional requirements.

While diagrams such as Combined Viewpoint Modelling are treated as part of a specification of a software system, they can not be included in specifications of instrument systems. Although decompositions help analysts gain insight into systems' functional behaviour, they must not form part of a specification document.

8. Conclusions

Some of the well-known structured specification methods have been briefly examined against a set of evaluation criteria, in the context of measuring instrument systems. Promising techniques include CORE, SCS and SREM. Due to CORE's explicit graphical notation, it provides systematic and effective support for functional analysis of complex instrument systems, for example, oscilloscopes, spectrometers, process control and instrumentation loop, and so on. Although decomposition details using the CORE notation should not be included in a specification of instrument systems, the analysis process helps to create complete and consistent functional specification. However, this advantage is not as obvious for simple instruments such as pressure sensors.

Prototype automated tools for the generation of measuring system requirements, using traditional checklisting approaches, are now available [Cook, 1988, 1990; Sydenham, 1990]. We are developing a prototype support environment for the specification of instrument systems, based on the proposed ViewPoint development framework [Finkelstein et al, 1990b]. This environment accommodates multiple representation and development techniques, covering both software and hardware engineering disciplines.

9. Acknowledgements

The authors wish to thank the SERC for the SEED research grant. They would also like to express their gratitude to Dr. S. Rozen, for his advice and criticism.

10. References

Alford M.W. 1980, Software Requirement Engineering Methodology (SREM) at the Age of Four, *Proceedings of the Int. Computer Software & Applications Conf.*, New York: IEEE Computer Society Press, pp.866-874

Balzer R. & Goldman N. 1986, Principles of Good Software Specification and their Implications for Specification Languages, *Software Specification Techniques* (ed. Gehani, N.H. & A.D. McGettrick)

Birrel N.D. & Ould M.A. 1985, *A Practical Handbook for Software Development*, Cambridge University Press

Coad P. & Yourdon E. 1991, *Object-Oriented Analysis (2nd Edition)*, Yourdon Press

Cohen B., Harwood W.T. & Jackson M.I. 1986, *The Specification of Complex Systems*, Addison-Wesley

Cook S.C. 1988, Automatic Generation of Measuring Instrument Specifications,

Measurement, Vol.6, No.4

Cook S.C. 1990, *A Knowledge-Based System for Computer-Aided Generation of Measuring Instrument Specifications*, PhD Thesis, City University, London

DeMarco T. 1978, *Structured Analysis and Systems Specification*, Yourdon Press

Finkelstein A.C.W. & Potts C. 1987, Building Formal Specifications Using Structured Common Sense, *Proceedings of 4th International Workshop on Software Specifiation and Design*, Monterey, California, USA

Finkelstein A.C.W., Maibaum T. & Finkelstein L. 1990a, Engineering-in-the-Large: Software Engineering and Instrumentation, *Proceedings of UK IT 1990*, Southampton, UK

Finkelstein A.C.W., Kramer J. & Goedicke M. 1990b, Viewpoint Oriented Software Development, *Proceedings of the Third International Workshop on Software Engineering and its Applications*, Toulouse, France

Finkelstein L. & Finkelstein A.C.W. 1987, Instruments and Instrument Systems: Concepts and Principles, *Systems & Control Encyclopedia* (editor: M.G. Singh), Pergamon Press, pp2527-2533

Jackson M.A. 1983, *System Development*, Prentice-Hall: London

MIL-STD-490A 1985, *Military Standards: Specification Practices*, United States of America Department of Defense

Nuseibeh B.A. 1989, *CoreDemo: An Investigation into the Use of Object-Oriented Techniques for the Construction of CASE Tools*, MSc Thesis, Imperial College, London

Sage A.P. (ed.) 1977, *Systems Engineering: Methodology and Applications*, IEEE Press, New York

Sydenham P.H., Harris D.D. & Hancock N.H. 1990, MINDS - A Software Tool to Establish a Measuring System Requirement, *Measurement*, Vol. 8 No.1

Systems Designers 1985, *CORE - the Method*, User Manual

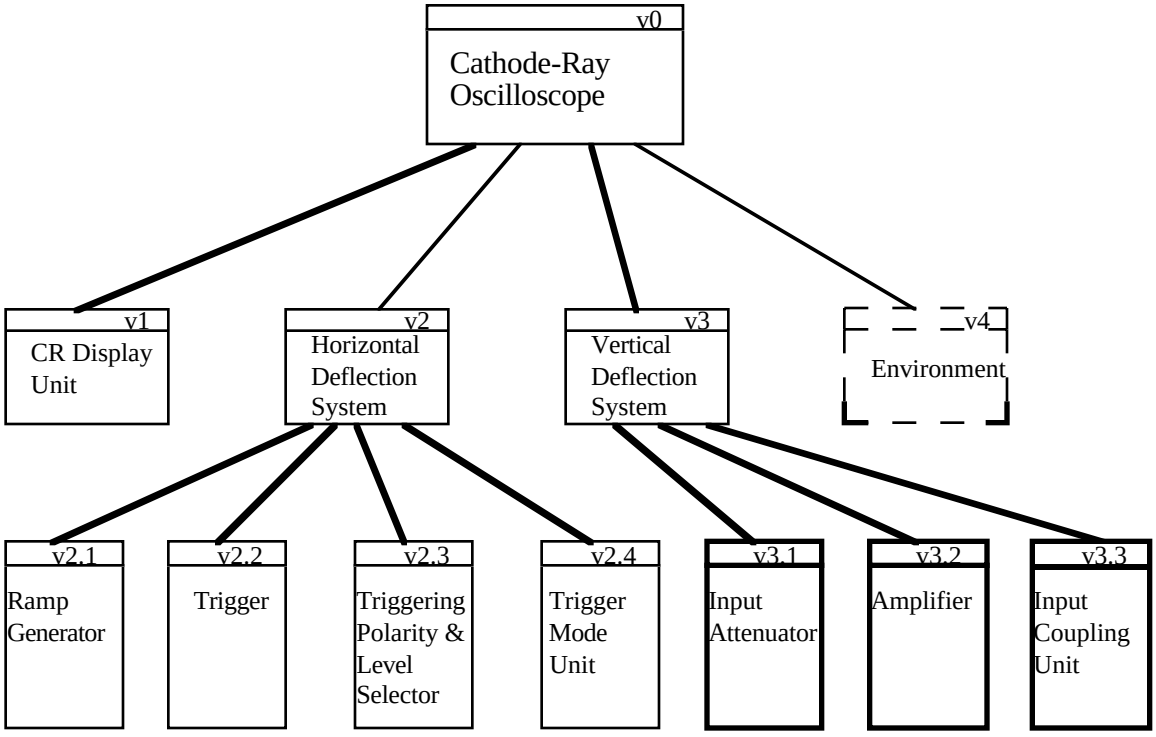

Figure 1. Viewpoint Structuring

V3.4 Trigger_Mode_Unit

Source	Input	Action	Output	Destination
Amplifier Environment	Amplified input signal External signal Line signal	Select & Couple signal	Coupled_signal	Triggering Polarity & Level Selector

V3.3 Triggering_Polarity_&_Level_Selector

Source	Input	Action	Output	Destination
Trigger Mode Unit	Coupled_signal	Select triggering polarity & level	Trigger_signal	Trigger

Figure 2. Tabular Collection

Outputs of Trigger_Coupling_Unit

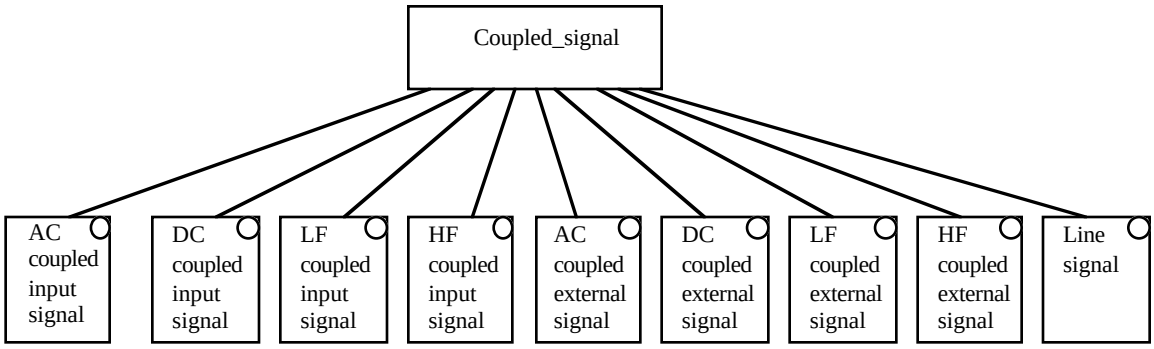

Figure 3. Data Structuring

V3.4 Trigger_Mode_Unit

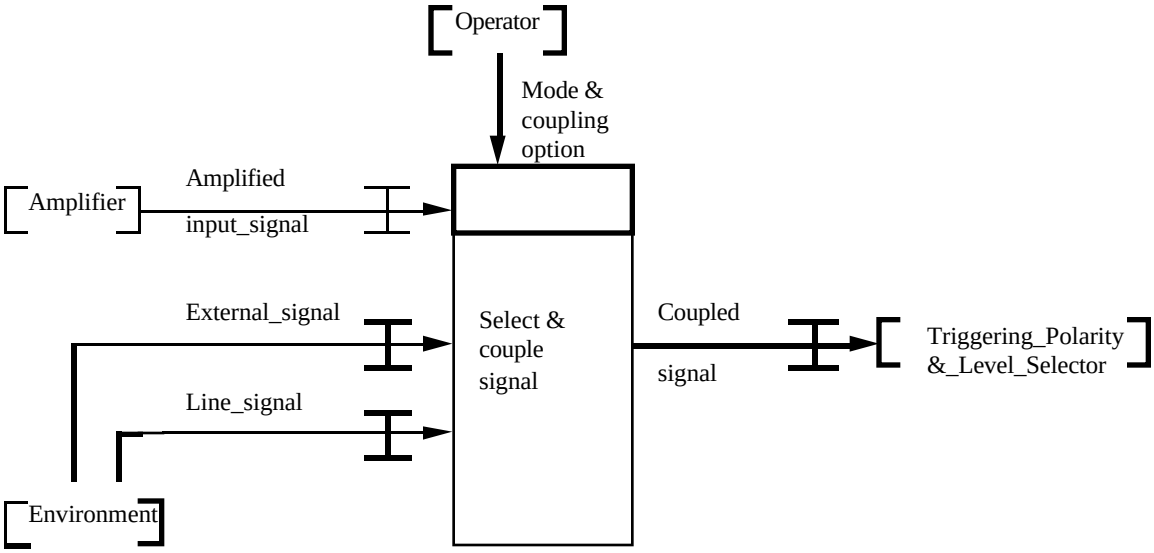

Figure 4. Single Viewpoint Modelling

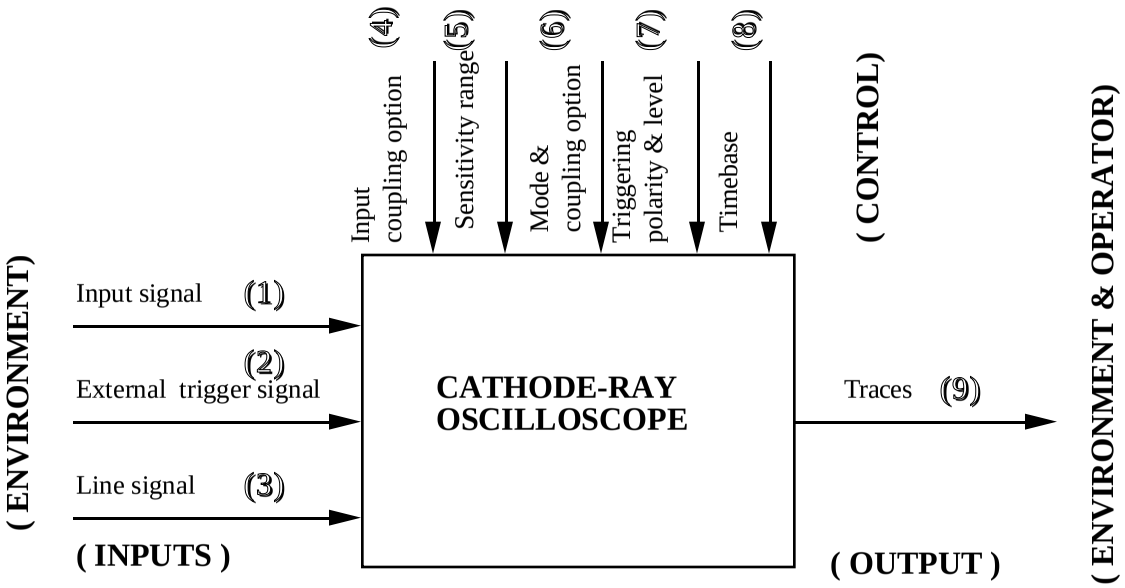

Figure 6. Functional behaviour of the oscilloscope