(In) Measurement Journal, Vol 10, No 2, Apr-Jun 1992, pp.87-92.

Using Software Specification Methods for Measurement Instrument Systems

Part 2: Formal 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

In the second part of the paper, we investigate the applicability of formal methods to the specification of measuring instrument systems. We then conduct a case study in the widely used Z method. Using formal methods for specification purposes, one can obtain a clear understanding of user's problems, especially the aspects of functional behaviour, and therefore produce a correct specification document based on this understanding.

Keywords: Requirements analysis, Formal specification, Z, Measuring instrument systems

1. Introduction

Formal specification methods have aroused great interest among the software community [Hall, 1990]. However, little is known about the applicability of formal methods to the specification of measuring instrument systems. In this part of the paper, we investigate this area, which has been almost totally ignored by the instrument community.

A client usually gives a requirement definition document, and a system analyst must then analyse these user requirements and translate them into a specification, which, once approved by both the client and the supplier, actually forms a legal document binding both sides. Therefore, the specification process consists of two important stages: requirements analysis and specification documentation. The analysis of user requirements leads to a final set of statements in a specification document. Some have used the term "formal specification" to refer to a strictly mathematical interpretation of user requirements in a specification document [Delisle and Garlan, 1990]. To avoid confusion, we use the term "specification document", instead of "specification". We discuss the prospect of applying formal methods to instrument specification from the viewpoints of both requirements analysis and specification documentation.

2. Major catagories of formal specification techniques

In model-based methods such as Z and VDM, a specification document is an explicit system model constructed out of abstract or concrete primitives. All the primitives are well defined. The existence of a model ensures consistency of the specification and gives useful hints to designers.

Algebraic specification of a certain data type uses axioms to describe the relationship (usually equality) between various operations that can be performed on this type of data.

Logic programming in, say, Prolog can be used to write specification documents by using a restricted form of first-order logic (Horn clauses) which can be reasoned about by adopting a resolution theorem proving approach. Even second-order predicates can be interpreted in Prolog.

These different styles of specification methods may be used for different types of specification problems. Pure mathematicians may want to use axiomatic algebraic methods to specify operations in terms of their syntax and semantics [Gehani, 1986]. However, these methods do not accommodate behaviour specification, e.g. input-output functional characteristics. A model-based approach such as Z or VDM is more applicable in this respect. Therefore, programmers and specifiers of instrument systems may prefer such an approach to an algebraic method. Due to good software support and reasoning capabilities available in a Prolog environment, some may choose to use Prolog to generate a specification document [Cook, 1990].

In the rest of this paper, we will investigate the applicability of model-based approaches to the specification of instrument systems.

3. The behaviour of software and instrument systems and implications for formal specification methods

All software behaviour can be modeled as a logical relation between inputs and outputs [Blackburn, 1989]. A strikingly similar approach to the modelling of instrument system behaviour was established by Finkelstein [1987]. In this approach, an instrument system is essentially a device which acquires and/or processes input and produces some form of output. Therefore, the functional behaviour of both software systems and instrument systems can be modeled as input-output transformations. The inputs and outputs can be considered to be sets of typed entities and the transformations as equality predicates. In the meantime, constraints on inputs, outputs and transformations can also be expressed using predicates. Therefore, model-based techniques may be applied to analyse the functional behaviour of software systems as well as instrument systems.

4. Benefits and Problems of Using Formal Methods

For software systems, model-based specification documents incorporate explicit system models, and they provide the basis for system designers to formally analyse the abstract models. Through such an analysis the designers can gain insight into the understanding of system requirements and obtain inspirations about design solutions. Although transformation details must not be included in a specification document of instrument systems, a formal analysis of customer requirements can lead to a clear and thorough understanding of user requirements, and therefore provide the basis for a correct interpretation of these requirements in a specification document.

A formal specification document of a software system also makes it possible to formally verify that the program meets its specification [Berg, 1982]. As a matter of fact, research into formal specification methods was initiated because of the need in formal program verification. Computer programs may be verified against their specification with some effort. Recently, formal methods have also been used to specify electronic circuits and verify their implementation against the specification [Hoare and Gordon, 1992]. However, it may be difficult to formally prove that a largely analog instrument actually conforms to its specification.

Formal specifications provide the basis for automatic program generation. More and more research has been devoted into tools for automatic transformations from formal specification into programs [Partsch, 1983; Balzer, 1985]. Transformation of chip specifications into chip products can also be automated [Hoare, 1992]. This automation may, however, be difficult to realise for analog instruments.

The economic implications of formal specifications are not yet that clear. However, it is claimed [Hall, 1990] that formal specification may reduce the overall cost of software development. Although more time and labour are involved in the specification phase than usually, the implementation and testing phases can be shorter. Therefore, the use of formal specification can reduce the overall life-cycle cost. However, formal methods may be expensive to adopt for instrument system specification, as design, implementation and testing can not be formally conducted in instrument development, let alone the fact that engineers usually try to avoid using mathematics.

Despite the tremendous effort on formal specification techniques, there are serious obstacles which prevent system analysts from using these techniques. Perhaps most importantly for industrialists, analysts are usually not familiar with discrete mathematics and logic which are essential for formal requirements analysis and specification. A considerable amount of training must, therefore, be provided.

Several case studies of formal specification for real-world software systems have been performed [Hayes, 1987]. Some of the practical applications in software engineering which have proved successful include the specification of a UNIX filing

system [Morgan et al, 1984]. In the following section, we formally analyse a differential pressure sensor using the model-based Z method, and then produce a corresponding specification document.

5. Specification Case Studies Using Z

5.1. The Z notation

Before we undertake any formal analysis, the basic Z notation, as used in the case studies, is briefly explained below.

x: X	Declaration that x is a member of typed set X
P S	Power set of set S
$X \pfun Y$	The set of partial functions from X to Y
$\forall$ x: X | P1, $\exists$ y: Y $\bullet$ P2	The logical statement that for all x of type X satisfying predicate P1, there exists a y of type Y such that predicate P2 holds
$\land$	Conjunction
$\times$	Cartesian product
λx: X $\bullet$ f(x)	The function that, given an argument x of type X, gives f(x)
S $\dres$ R	Domain restriction: { (x,y) | (x,y) $\in$ R $\land$ x $\in$ S }
==	Equivalence

Schema definition:

Schema-Name

Declarations

Predicates

The Δ schema convention:

ΔS == S $\land$ S'

where S is the before state (schema) and S' is the after state (schema)

Axiomatic-box definition:

Declarations

Predicates

5.2 Specifying a Differential Pressure Sensor

Suppose the customer has given the following description of a differential pressure sensor: The device shall generate an analog signal output proportional to a pressure difference between atmospheric and input pressure and ratiometric to reference voltage, subject to performance and other constraints. Based on the above general description, we will analyse the functional transformation of the pressure sensor and non-functional constraints the sensor must conform to, giving a Z specification for the sensor. The following data types will be used in the analysis of input-output transformation and non-functional constraints:

mS, V, kPa, mA, kΩ, kHz: P R
(Time in mS, voltage in V, pressure in kPa, current in mA,
impedance in kΩ and frequency in kHz, are real numbers)

5.2-1 Functional transformation

The relationship between input pressure (p), supply voltage (e_R) and output voltage (e_o) must be clearly defined, say, as

e_o, e_R: V
p: kPa
$e_o = e_R * (0.01 * p + 0.1)$

The allowed range of supply voltage, e_R, shall be from 4.7 to 5.3, for example. There must also be an upper limit to the measurand, say, $p \leq 80$.

We can now use the following schema box to express the transformations of the pressure sensor. The variables followed by **?** represent input variables, and those followed by a **!** represent output variables.

PressureSensor

p**?**: kPa
e_R**?**, e_o**!**: V

$(4.7 \leq e_R? \leq 5.3)$
$(p? \leq 80)$
$(e_o! = e_R? * (0.01 * p? + 0.1))$

Alternatively, the function type in Z can be used to represent the input-output transformation of the sensor. The following axiomatic-box defines the transformation from (Differential_pressure $\times$ Reference_voltage) to Output_voltage.

$$\text{PressureSensor}: \text{kPa} \times \text{V} \rightarrow \text{V}$$

$$\forall e_R: \text{V} \mid 4.7 \leq e_R \leq 5.3; \text{p: kPa} \mid p \leq 80 \bullet \text{PressureSensor}(p, e_R) = e_R * (0.01 * p + 0.1)$$

5.2-2 Non-functional constraints

All the constraints which are not directly related to input-output transformations are regarded as non-functional constraints. We do not attempt to give an exhaustive list of such constraints. Typically, restrictions on current drain (c_d), dynamic time response (t_{res}), impedance (Imp) and sensor output error (er) will have to be specified.

c_d: mA
$c_d \leq 20$

t_{res}: mS (step response time as 90% of steady state value)
$t_{res} \leq 15$

Imp: kHz $\rightarrow$ kΩ (impedance as a function of frequency)
$\text{Imp}(1) \geq 10$

Sensor output error, *er*, shall be expressed as an equivalent change in the pressure measurand, *p*. We first specify the error over a critical operating temperature range, say, $t \in$ (0°C, 80°C). The error shall lie between the upper limit of (-0.05*p + 1.5) and the lower limit of (0.05*p - 1.5) if the differential pressure *p* is less than 10kPa; between 1.0 and -1.0 if $10\text{kPa} \leq p \leq 50\text{kPa}$; or between (0.05*p - 1.5) and (-0.05*p + 1.5) if $50\text{kPa} \leq p \leq 80\text{kPa}$. A factor of (-0.05*t + 1) can be multiplied to these bounds if temperature $t \in$ (-40°C, 0°C), or (0.05*t-3) if $t \in$ (80°C, 120°C). To specify such a complex constraint in Z, we first define a set of multiply factors, M, as a function of temperature, over the temperature range (-40°C, 120°C), as

$$\text{T: P R}$$
$$\text{M: T} \rightarrow \text{R}$$
$$\{t: \text{T} \mid -40 \leq t \leq 120\} \lhd \text{M} == \lambda t: \text{T} \mid -40 \leq t \leq 0 \bullet (-0.05*t + 1) \cup \lambda t: \text{T} \mid 0 \leq t \leq 80 \bullet 1 \cup \lambda t: \text{T} \mid 80 \leq t \leq 120 \bullet (0.05*t - 3)$$

The following predicate then defines the constraint on sensor output error *er*

$$\forall p: \text{kPa} \mid p \leq 80; t: \text{T} \mid -40 \leq t \leq 120, \exists er: \text{kPa} \bullet ((p \leq 10) \wedge ((0.05*p - 1.5) \leq er/M(t) \leq (-0.05*p + 1.5))) \vee ((10 \leq p \leq 50) \wedge (-1.0 \leq er/M(t) \leq 1.0)) \vee ((50 \leq p \leq 80) \wedge ((-0.05*p + 1.5) \leq er/M(t) \leq (0.05*p - 1.5)))$$

5.3 Functional Specification of a generalised chemical process model

A generalised chemical process control loop [Williams, 1960] is given in Figure 1. The plant considered here is to produce a certain amount of chemical product, P, whose exact nature need not be specified here. We will not consider the detailed chemical kinetics of the reactions involved in producing P. Dynamics of the process is ignored and steady state is assumed for all flows and reactions. This case study is mainly to demonstrate how Z can be applied to the specification of functional behaviour of complex instrument systems, instead of simple instruments such as pressure sensors.

Since flow rates (e.g. in lb/hr) and temperatures (e.g. in ^{o}C) will be used intensively in the following analysis, we first define them as real numbers

FR, T: P R

5.3-1 Reactor

The main chemical reaction of the process occurs in the reactor, which takes initial reactants A and B, and produces a mixture of various chemicals, including A, B and P. We denote the output of chemical mixture as f_{ri}. Note that a recycling flow of f_l also feeds into the reactor. A flow of water, $f_{w1,}$ is used for cooling purpose, with inward flow temperature t_{w1i} and outward flow temperature t_{w1o}.

The state of the reactor can be specified as

Reactor

f_{w1}: FR
$t_{w1i,}$ t_{w1o}: T

and the operation "React" is specified by

React
Δ Reactor
$f_a?$, $f_b?$, $f_l?$, $f_{ri}!$: FR

$f_{w1}' = f_{w1}$
$t_{w1i}' = t_{w1i}$
$f_{ri}! = f_a? + f_b? + f_l?$

which simply implies that a constant flow rate of coolant water and constant temperature of inward water flow, must be maintained before and after the reaction. The flow rate of

output, f_{ri}, is to be the sum of all input flow rates, i.e. f_a, f_b and f_l.

5.3-2 Reaction-Cooler Heat Exchanger

The reaction-cooler heat exchanger is by name to reduce the temperature of chemicals produced by the reactor. A heavy oil waste material, G, was produced after the initial reaction and it must be made insoluble through a cooling operation. Another purpose of the cooling procedure is to effectively stop the reaction, thus avoiding an overproduction of G. The state space of the heat exchanger can again be considered as including the flow rate of cooling water, f_{w2}, inward flow temperature t_{w2i} and outward flow temperature t_{w2o}.

ReactionCoolerHeatExchanger

f_{w2}: FR

t_{w2i}, t_{w2o}: T

The operation "Cool Reaction" is described by

CoolReaction

Δ ReactionCoolerHeat Exchanger

$f_{ri}?$, $f_r!$: FR

$f_{w2}' = f_{w2}$

$t_{w2i}' = t_{w2i}$

$f_r! = f_{ri}?$

5.3-3 Decanter

Since G has a considerably higher gravity than the carrier stream, it may be removed by settling in a decanter. No specific state variables are associated with the decanter, and we only specify the decanting operation by the following schema

Decant

$f_r?$, $f_e!$, $f_g!$: FR

$f_g! + f_e! = f_r?$

5.3-4 Distillation Column

The final product, P, is separated from the carrier stream from the decanter, by distillation. We do not attempt to indulge ourselves deeply in the design of a distillation

column here. However, since the coolant water temperature is closely related to plant capacity to produce P, we will have to restrict the allowed range of the coolant temperature, say, below 10 ^{o}C. Let us assume that a stream rate of 5000 lb/hr of P shall be produced from the plant. The design of the distillation column will depend on the maximum allowed coolant temperature and the required production rate of P. The state space of the distillation column is given as

DistillationColumn

f_{w3}: FR

t_{w3i}, t_{w3o}: T

$t_{w3i} \leq 10$

The distillation operation can be specified as

Distill

Δ DistillationColumn

$f_e?, f_p!, f_s!$: FR

$f_{w3}' = f_{w3}$

$t_{w3i}' = t_{w3i}$

$t_{w3i} \leq 10$

$f_p! + f_s! = f_e?$

$f_p! = 5000$

5.3-5 Recycle Control

After the distillation operation, the column bottom stream, f_s, still contains some amount of P. A certain proportion of this stream is discarded, while the rest is to be returned to the reactor for recycling. The proportion, f_d/f_s, may vary from one designer to another. The overall proportioning control may be described as

FlowProportioningControl

$f_s?, f_l!, f_d!$: FR

$f_l! + f_d! = f_s?$

5.3-6 Functional specification of the overall process

The functional behaviour of the whole process may now be analysed, using the above descriptions of individual components:

$$Process == React \wedge CoolReaction \wedge Decant \wedge Distill \wedge FlowProportioningControl$$

Expansion of the schema Process gives the following:

Process

$f_{w1},\ f_{w1}',\ f_{w2},\ f_{w2}',\ f_{w3},\ f_{w3}' : FR$

$t_{w1i},\ t_{w1i}',\ t_{w1o},\ t_{w1o}',\ t_{w2i},\ t_{w2i}',\ t_{w2o},\ t_{w2o}',\ t_{w3i},\ t_{w3i}',\ t_{w3o},\ t_{w3o}' : T$

$f_a?,\ f_b?,\ f_{ri},\ f_r,\ f_e,\ f_s,\ f_l,\ f_g!,\ f_p!,\ f_d! : FR$

$f_{w1}' = f_{w1} \wedge f_{w2}' = f_{w2} \wedge f_{w3}' = f_{w3}$

$t_{w1i}' = t_{w1i} \wedge t_{w2i}' = t_{w2i} \wedge t_{w3i}' = t_{w3i}$

$t_{w3i} \leq 10$

$f_a? + f_b? + f_l = f_{ri}$

$f_{ri} = f_r$

$f_r = f_g! + f_e$

$f_e = f_p! + f_s$

$f_s = f_l + f_d!$

$f_p! = 5000$

In the above expansion, some variables, e.g. f_r, appear both as an input variable (decorated with **?**) and an output variable (decorated with **!**). Therefore, decorations have been removed to yield an undecorated variable. After further substitutional simplification, we have the following schematic specification of Process:

Process

$f_{w1},\ f_{w1}',\ f_{w2},\ f_{w2}',\ f_{w3},\ f_{w3}' : FR$

$t_{w1i},\ t_{w1i}',\ t_{w1o},\ t_{w1o}',\ t_{w2i},\ t_{w2i}',\ t_{w2o},\ t_{w2o}',\ t_{w3i},\ t_{w3i}',\ t_{w3o},\ t_{w3o}' : T$

$f_a?,\ f_b?,\ f_g!,\ f_p!,\ f_d! : FR$

$f_{w1}' = f_{w1} \wedge f_{w2}' = f_{w2} \wedge f_{w3}' = f_{w3}$

$t_{w1i}' = t_{w1i} \wedge t_{w2i}' = t_{w2i} \wedge t_{w3i}' = t_{w3i}$

$t_{w3i} \leq 10$

$f_a? + f_b? = f_g! + f_p! + f_d!$

$f_p! = 5000$

---

which can be interpreted into the following statements:

1. The plant is to use chemicals A and B to manufacture chemical product P at a rate of 5000 lb/hr.

2. During the manufacture of chemical P, a heavy oil material G will be

produced and it must be disposed of as a waste material.

3. Apart from product stream P and waste material stream G, there must be a channel for stream D to be discarded.

4. All coolant flowing into the plant must has a constant flow rate and they must be maintained at a constant temperature. In addition, the temperature of the coolant water used in distillation must not exceed 10 ^{o}C.

6. Conclusions

Using formal methods for specification purposes, one can obtain a clear understanding of user's problems, especially the aspects of functional behaviour, and therefore produce a correct specification document based on this understanding. The case studies presented in this paper have demonstrated this point very well. In the meantime, formal verification of specification consistency may be checked using formal proof methods. The task of such verification will not be pursued in this paper.

Formal specifications also provide the basis for formal validation checking and automatic specification-to-product transformation. This advantage is most obvious in software development and digital chip manufacture, though much remains to be explored of formal validation and automatic transformation in measurement instrument development.

7. Acknowledgements

The authors wish to thank the SERC for the SEED research grant. They would also like to express their gratitude to Dr. J. Moffett of Imperial College, for his criticism with respect to Z specifications.

8. References

Balzer R. 1985, A 15 Year Perspective on Automatic Programming, *IEEE Transactions on Software Engineering*, Vol.11 No.11

Berg H.K., Boebert W.E., Franta W.R. & Moher T.G. 1982, *Formal Methods of Program Verification and Specification*, Prentice-Hall International

Blackburn M.R. 1989, Using Expert Systems to Construct Formal Specifications, *IEEE Expert*, Vol. 4 No.1

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

Systems, Addison-Wesley

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

Delisle N. &Garlan D. 1990, A Formal Specification of an Oscilloscope, *IEEE Software*, Vol.7 No.5

Finkelstein L. 1987, Instrument and Instrument Systems: Concepts and Principles, *Systems and Control Encyclopedia* (editor: M.G. Singh), Pergamon Press

Gehani N.H. & McGettrick A.D. (ed.) 1986, *Software Specification Techniques*, Addison-Wesley, pp.173-185

Hall A. 1990, Seven Myths of Formal Methods, *IEEE Software*, Vol.7 No.5

Hayes I. (ed.) 1987, *Specification Case Studies*, Prentice-Hall International

Hoare C.A.R. & Gordon M.J.C. (eds.) 1992, The Royal Society Discussion Meeting on *Mechanized Reasoning and Hardware Design*, to be published by Prentice-Hall International in June 1992

Morgan C. & Sufrin B. 1984, Specification of the UNIX Filing System, *IEEE Transactions on Software Engineering*, Vol.10 No.2

Partsch H. 1983, Program Transformation Systems, *ACM Computing Surveys*, Vol.15 No.3

Sommerville I. 1989, *Software Engineering*, Addison-Wesley

Williams T.J. & Otto R.E. 1960, A Generalized Chemical Processing Model for the Investigation of Computer Control, *IEEE Trans. on Communications and Electronics*, Vol.79

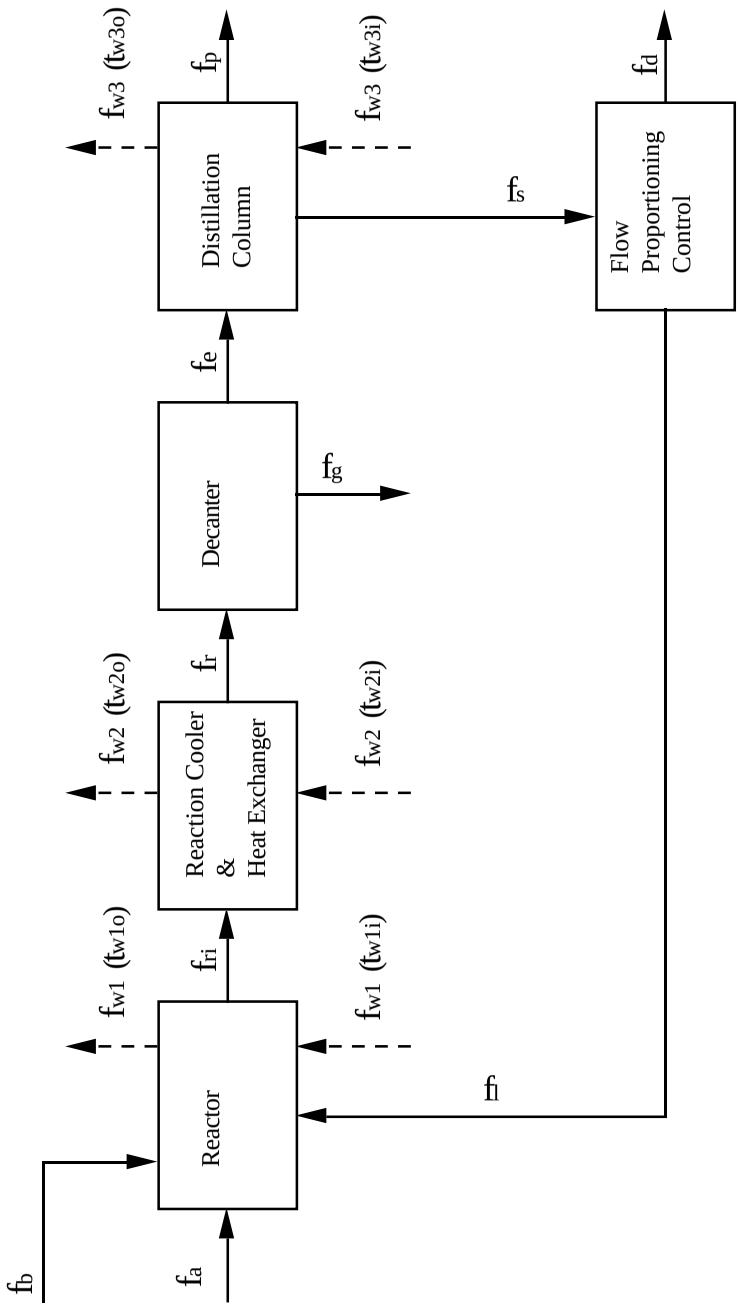

Figure 1. A Generalised Chemical Process Model for Control