Volume 7, Issue 4, April 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Troubles Construct Substance for Package Growth

[1]Shaista Sabeer, [2]Ayasha Siddiqua, [3]Nivedita Soni
[1]Lecturer Jazan University, Saudi Arabia
[2] Lecturer Jazan University, Saudi Arabia
[3]Technical Analyst HCL

Abstract: ***This research paper demonstrates a model for realizing Trouble Forms. The substance directs Trouble Fleshes within the model for Demands Engineering of Zave and Jackson, and its later formalization in the Reference Model of Gunter et al. It discerns among trouble frames, context plots and trouble plots, and appropriates us to officially determine the relationship among them as assumed in the Trouble Frames model. The semantics of a trouble plot is given in terms of 'challenges', a notion that we also introduce. The notion of a challenge is interesting in its own right for two reasons: its proof theoretic derivation leads us to consider a challenge calculus that might underpin the Trouble Frame operations of fragmentation and constitution; and it predicts to cover the notion of formal elaboration from package growth to necessities technology. In addition, the semantics supports a textual representation of the plots in which Trouble Frames catch troubles and their relationship to solutions. This could open the way for graphic Trouble Frames tools.***

Keywords: Demands Engineering, Trouble Frames, Meaning, Reference Model, Model

I. INTRODUCTION

Trouble Frames [8, 9] relegate package growth troubles. In their purpose they are cognate to invention patterns [4, 10], but for depicting troubles, not results. Trouble Frames structure the world in which the trouble is settled—the trouble field—and depict what is there and what consequences a arrangement located there must accomplish. By underlining troubles rather than solutions, Trouble Frames can exploit the understanding of a trouble class, allowing a trouble owner with particular field knowledge to drive the Necessities Engineering process. The graphic notation is cooperative here for communicating among the package programmer and the trouble owner.
However, a mainly graphic basis has disadvantages [14]. Particularly:

- It is difficult to understand graphical artifacts exactly, and mistakes can well occur;
- It is difficult to influence whether a verbal description is concluded and correct;
- It is difficult to distinguish equivalent structures that could be employed interchangeably.

Other graphical notation districts have found that semantics leads to a solution, see [7, 11] and research papers in [3] for cases. The semantics confronted in this research paper could attend this function for Trouble Frames. To the best of our cognition, semantics of Trouble Frames is lacking from the literature, although a partial formal depiction of some early work on Trouble Frames is given in [1].
Our semantics identifies Trouble Frames within Zave and Jackson's model for Requirements
Engineering [15]. Particularly we center on the relationship between Trouble Frames and expressions of the form

$$K, S \vdash R$$

(Where K is a verbal description of the trouble field, S is the stipulation of the result, and R is the trouble requirement). This relationship associates the Trouble Forms model to the Reference Model for necessities and perticularations given by Gunter et al [5] and more lately, Hall and Rapanotti [6], in which assure and other interaction places are given expression.

The meaning has many concerning characteristics. For instance:

- It constitutes the structure of a Trouble Frames artifact at a level over the detail of field descriptions; for instance, it can adapt the verbal description of an 'operator' field's behaviour as
- 'The operator bullets'.
- In this sense, our delegacy of the Trouble Frame artifacts arrives from the tradition of the propositional calculus: we represent and manipulate only the structure above the level of field descriptions. Of course, as with the propositional calculus—from which the predicate calculus is reached through the formalisation of propositions—so too, in principle, our semantics is extensible wherever more formal field descriptions exist.
- We define a formal syntax for Trouble Frames, underpinning the semantics and anchoring the graphical notation. This syntax opens the way for graphical Trouble Frames tools and for electronic transformation and communication of Trouble Frames artefacts.

- The intending of a trouble plot is given in terms of 'disputes'—a formal notion that abstractions the answer S from K, S ` R tuples. The impression of a challenge is interesting in its own right for at least two concludes. Its proof theoretical derivation leads us to conceive a challenge calculus that might underpin the Trouble Frame operations of reaction and recomposition, as well as the linked correctness arguments; and it predicts to extend the notion of (formal) improvement from program growth through to requirements engineering.
- The semantics for Trouble Frames furnishes a clear overall structure for artefacts and their human relationship within a Trouble Builds growth. Although persuasive debates for the soundness of a growth can be based exclusively on its rigour or formality, a clear overall structure is invariably the necessity basis. Without a clear overall structure, even a aggregation of utterly formal and converting arguments at the detailed level will not add up to a presentment that a growth is sound.

This research paper is formed as follows. Section 2 outlines the TF (Trouble Frames) model. A Comprehensive introduction to TFs is beyond the scope of this research paper and can be found in [9]. Section 3 remembers the Reference Model of [5]. Section 4 brings in the semantics. Finally, Section 5 in short points out on the nature of the semantics and shows the direction of future work.

II. THE TROUBLE MODEL

In this section we brush up some of the basic elements of the TF (Trouble Frames) model.
To concenter our brush-up we depict the TF growth of the following trouble, a Chemical Reactor Controller:
A computer system is commanded to assure the catalyst unit and cooling of a chemical reactor. An operator consequences commands for activation or deactivating the catalyst unit; in response to such commands, the system teaches the unit consequently and governs the flow of cooling water. A gearbox is accompanied the system: whenever the oil level in the gearbox is low, the system should ring a bell and halt performance.
This trouble is a altered version of a chemical reactor depicted in [2, 12].

2.1 Trouble Plots

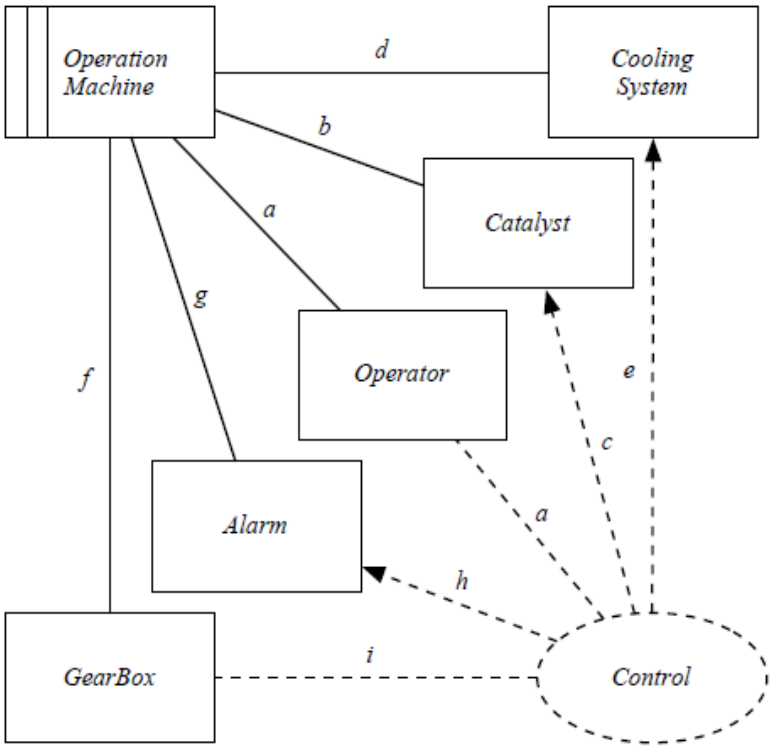
Within the TF model, a trouble plot determines the 'shape' of a trouble by catching the Characteristics and interconnections of the elements of the world it is concerned with. A trouble plot also admits the requirements that constrain the relationships between these elements. The trouble plot for our Chemical Reactor Restrainer trouble is demonstrated in Figure 1.

The elements are:

- A double-blocked box (Operation Machine): the automobile field, i.e., the package arrangement to be built collectively its fundamental hardware.
- A box (the Operator field): the human manipulator. Human manipulators are looked upon acquiescent in the trouble forms model: they may obey specified procedures, but not constantly, and may give events spontaneously.
- Other boxes (Cooling System, Catalyst, etc.): given areas constituting elements of the world that is applicable to the trouble. Each of these areas is causal: i.e., its phenomena are forcible events and states, and are causally related;
- A dotted oval (Control): the necessity: i.e., the circumstance in the trouble field that the Operation Machine must assure to characterize as a result to the trouble.
- The commented associations among areas: these suggest shared developments, accepting events, operations and state data. In Figure 1, for instance, the association among the Operation Machine and the Cooling System is commented by the set d containing the two phenomena increase wateract and diminish wateract. These constitute commands rendered by the Operation Machine in order to gain or decrease the water level in the Cooling System. Phenomena divided among two fields are observable by both, but checked by one of them only. For instance developments gain wateract and decrease wateract are both controlled by the Operation Machine: this is suggested by an form of the field name followed by!, in this case OM!.
- The footnoted associations among the requirement and the fields: a dashed line proposes that the necessity references its phenomena; while a damned arrow points that the necessities constrain the phenomena. For instance, the oil level in the gear case is cited by the requirement, while the bell ringing is encumbered.

As well as acquiescent and causal, fields can also be lexical; a lexical area reifies a data structure, in some cases a SMS. The phenomena of biddable fields are events; those of causal fields are nations and effects (generalized as causal phenomena); and those of lexical fields are symbolic.

Not accepted in the plot, but a necessity part of the trouble result, are the description of all given areas and the details of the necessity. Typically, as the growth encouraged, point of a machine stipulation would be added. All of these descriptions would be enrolled elsewhere, rather than added to the plot. A significant point is that the model does not dictate the notation to be employd for these descriptions; formal and informal descriptions are evenly acceptable, furnished they are precise enough to catch the features of interest of the various respective fields. This freedom prompts the suggestion nature of our semantics: the expiation of a description, for example by a field, can be absentminded as a proposition.



$a : OP!\{open\_catalyst, close\_catalyst\}$
$b : OM!\{open\_catalyst_{act}, close\_catalyst_{act}\}$
$CA!\{is\_open_{sen}, is\_closed_{sen}\}$
$c : CA!\{open, closed\}$
$d : OM!\{increase\_water_{act}, decrease\_water_{act}\}$
$CS!\{is\_rising_{sen}, is\_falling_{sen}\}$
$e : CA!water\_level$
$f : GB!request\_service$
$g : OM!ring\_bell$
$h : AL!bell\_ringing$
$i : GB!oil\_level$

Figure 1. The Apparatus Trouble Plot

The TF model discerns two fundamental roles of descriptions:

- Declarative, conveying what is given or assumed to be true, and
- Optative mood, conveying what is required or desired to be true.

Descriptions of given fields, then, play an indicative role, while the necessity description and machine stipulation are optative. Also furnished separately from the plot are the designations of phenomena, which foundation the trouble's lexicon in the physical henomena. For instance, through their recognitions, open accelerator and block catalyst will denote forcible Control actions that the operator can actually execute; is ascensions and is fallings will denote states that can be caught by sensors.

2.2 Trouble Frames

One of the directs of the TF model is to depict basic classes of trouble that repeat throughout package growth. [9] depicts five such basic trouble categories, or trouble frames, accepting the checked behavior frame, the commanded behaviour frame and the data display frame, which we employ here (or later) for example. A trouble frame plays a guide for distinguishing difficulties in its class. In form a trouble frame is merely a trouble plot, commented with field and phenomena gradings, and linked with a correctness argument template. To match a trouble frame template, a trouble must:

- match the trouble topology—as captured in the trouble frame plot;
- match the field characteristics—as captured by the field markings in the trouble frame plot (B for biddable, C for causal, and Y for lexical);
- match the characteristics of the shared phenomena—as captured by the markings on the connections (E for events, C for causal and X for symbolic); and
- Support a way of building and discharging a correctness argument which matches the correctness argument template for the trouble class. For the commanded behaviour frame, the form of the debate will exploit explicitly stated causative places of the assured field to demonstrate that the machine behavior in terms of the phenomena E4, C1 and C2 will case the commanded behaviour of the controlled field in terms of the phenomena C3. The reader concerned in the detail of the correctness argument template should consult [9].

The controlled behaviour form is demonstrated as the annotated trouble plot in Figure 2.

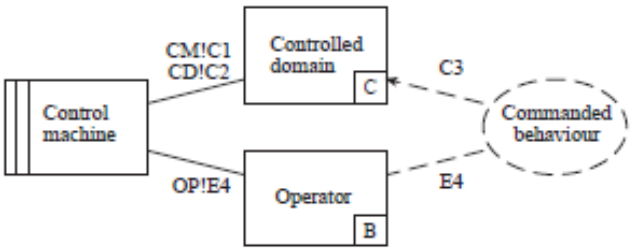


Figure 2. The Commanded Behaviour frame

By equating Figure 2 and Figure 1, the proofreader will see that the controlled behaviour frame matches the ejection of the trouble plot comprising only of these elements:

- The Operation Machine, matching the Control Machine in the construct.
- The causal Catalyst, corresponding the Controlled field in the frame.
- The biddable Operator, corresponding the Operator in the frame. The operator controls and consequences commands spontaneously as event phenomena, shared with the Control machine. These event phenomena are open catalyst and close catalyst.
- The requirement determining the response of the Controlled field to the Operator's commands. The required behaviour is expressed in terms of the causal phenomena is opensen and is closedsen.

2.3 Trouble Rotting

Most real troubles are too composite to fit basic trouble frames. They require, rather, rotting into a collection of acting sub-troubles, each of which is smaller and merer than the original. The projection of the trouble plot of Figure 1 that matches the commanded behaviour frame represents one such sub-trouble.

Within the TF model, we depict sub troubles from their trouble frame guides, and instantiate the corresponding sub-trouble plots. Our substance appropriates us to express the relationship among troubles and sub-troubles; we will in short illustrate this process for the chemical reactor.

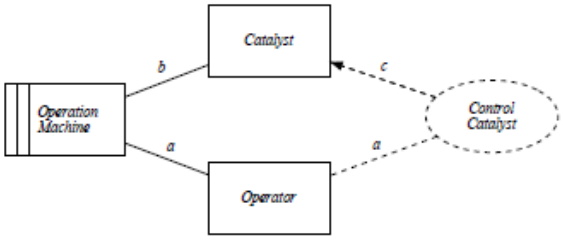
The chemical reactor trouble can be broke down into three distinct sub-troubles:

- A commanded behaviour frame allowing the operator to control the catalyst;
- A required behaviour frame regulating the water flow for cooling; and
- An information display frame for issuing a warning (and halting the system) when there is an oil leak in the gearbox.

To be able to apply the trouble frames as guides, the fields and phenomena of the trouble must correspond those of the frame. The delegacy of a trouble frame for a particular trouble results in the sub-trouble plot that was corresponded by the trouble frame. In the Chemical Reactor trouble, the representation of the commanded behaviour frame is the trouble plot of Figure 3; that of the required behaviour frame is given in Figure 4; and that of the information display frame in Figure 5.

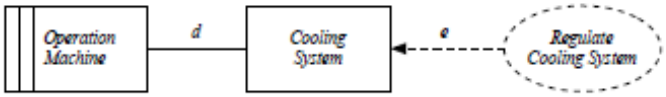
III. THE ACKNOWLEDGMENT MODEL

In the division we in short remember the Reference Model, which underpins our semantic delineation of Trouble Frames. The Reference Model [15, 5, and 6] is demonstrated on the widely accepted separation among the system (Which we call the machine) and its surroundings (which we call the trouble fields); on a number of key artifacts which pertain to the two; and on a vocabulary, which is wont to depict the environment, the system and the employr interface among them. From a necessities engineering viewpoint, the key artifacts are the following1: the field knowledge, K, is what we know about the surroundings; the requirement, R, is what the customer involves of a system working within the surroundings; and the perticularation, S, is a description of the system, which can be employd for its implementation.



$$a : OP!\{open\_catalyst, close\_catalyst\}$$
$$b : OM!\{open\_catalyst_{act}, close\_catalyst_{act}\}$$
$$CA!\{is\_open_{sen}, is\_closed_{sen}\}$$
$$c : CA!\{open, closed\}$$

Figure 3. The Control Accelerator sub trouble, an instantiated controlled behaviour frame

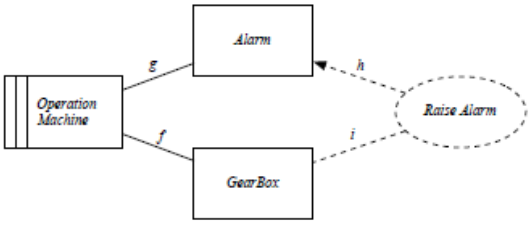


$$d : OM!\{increase\_water_{act}, decrease\_water_{act}\}$$
$$CS!\{is\_rising_{sen}, is\_falling_{sen}\}$$
$$e : CA!water\_level$$

Figure 4. The Regulate Cooling System sub trouble, an instantiated commanded behaviour frame

The Reference Model validates the relation [15] among world, demands and machine as:

$$K, S \vdash R$$



$$f : GB!request\_service \quad g : OM!ring\_bell$$
$$h : AL!bell\_ringing \quad i : GB!oil\_level$$

Figure 5. The Raise Alarm sub trouble, an observed data display frame

That is: a proof can incline that an effectuation of S, introduced into a trouble field meeting K, will ensure satisfaction of the requirement R.

In the Reference Model, appointments are employd to basis terms in the real world [15]. The appointments furnish names to depict the environment, the system and their artefacts. The vocabulary is wont to name phenomena, which are grouped into: those announced by e, which belong to, and are assured by, the environment; and those, announced by s, which belong to, and are controlled by, the system (or machine). e is further partitioned off into phenomena which are seeable to (or observable by) the machine (ev), and those that are concealed from it (eh); s is further partitioned off into phenomena which are seeable to (or observable by) the environment(sv), and those that are hidden from it (sh). ev [sv is the vocabulary of the interface among the environment and the machine.

For technical grounds, the Reference Model presumes that the perticularation, S, is written in the

Interface vocabulary: i.e., S can employ only terms announcing phenomena in ev U sv . On the other hand, K and R can employ terms denoting phenomena in e U sv.

IV. TOWARDS SEMANTICS FOR TROUBLE FRAMES

In this section we depict the semantics in terms that we have already brought in. The semantics starts with a non-graphical representation language for the elements of the TF model. This is based on the (uninterpreted) Field and Demands Description Language (DRDL) which is wont to express necessities and field properties—our equivalents of basic suggestions. We then build trouble plots and trouble frames upon this, in the Trouble Frames Description Language (TFDL). Then we determine trouble frame instantiation, and finish with the interpreting of rotting and recomposition.

4.1 TFDL: a Description Speech for Trouble Frames

We bring in a textual annotation for trouble frames, the Trouble Frames Description Language (or TFDL). The function of this annotation is to provide a (textual) syntactic field on which to define our semantic purpose.

As talked about earlier, we presume that some language (or a collection of languages) has been preferred for the description of fields, phenomena, necessities and perticularations—these are the ‘basic propositions’ of our meanings. This we concern to as the Field and Necessity Description Language (or DRDL, for short). We presume only that DRDL:

- allows the delegacies of phenomena and their relationships, can discern controlling influence over them from reflection of their values, and can mark phenomena and fields appositely;
- Comes furnished with some notion of concluding, which we will pertain to as `DRDL (or simply as ` when the context makes this clear).

We do not control the concluding of `DRDL to be formal in any proof theoretic sense. It may be, e.g., that the concluding takes a pragmatic package technology form, such as testing. We return to this point in Section 5.

4.1.1 Depicting trouble plots

The basic elements of trouble plots are fields (including machine fields), necessities and their property through arcs and annotations. To be able to express this in TFDL, we presume the following sets, representing, respectively, all fields, all machine fields, all necessities and all phenomena:

PFDomain, PFMachine,
PFRequirement, PFPhenomena

For the grounding of a problem, both domains and requirements must have names and descriptions in $DRDL^2$:

PFDomainName
≙ PFDomain → *DRDL*
PFDomainDescription
≙ PFDomain → *DRDL*

PFRequirementName
≙ PFRequirement → *DRDL*
PFRequirementDescription
≙ PFRequirement → *DRDL*

Phenomena are also named and described in the *DRDL*:

PFPhenomenaDesignation
≙ PFPhenomena → *DRDL*

Phenomena are partitioned in sets of phenomena that can be either controlled or observed by each domain and the machine:

PFControlled
≙ PFDomain ∪ PFMachine → PFPhenomena
PFObserved
≙ PFDomain ∪ PFMachine → PFPhenomena

Similarly, some developments can be either restrained or referenced by the necessity:

PFConstrained
≙ PFRequirement → PFPhenomena
PFReferenced
≙ PFRequirement → PFPhenomena

We will packet all annotating operates into a trouble plot annotating:

PFAnnotation≙
(PFDomainName, PFDomainDescription,
PFRequirementName,
PFRequirementDescription,
PFPhenomenaDesignation,
PFControlled, PFObserved,
PFConstrained, PFReferenced)

(For briefness, if A is the trouble plot annotation, we will refer to the nine annotating purposes as A1, . . . ,A9, where A1 is the field names notation, A2 is the field descriptions annotation, and so on.) A trouble plot is a set of fields, a single simple machine, a single (set of) requirement(s), and an annotation, i.e.:

PFProblemDiagram≙
(ℙ PFDomain, PFMachine,
PFRequirement, PFAnnotation)

The agreement among the graphical elements of a trouble plot and their TFDL similitude is instanced in Figure 6. As an example, let ({D1, . . . ,D6},M,R,A),where A = (A1, . . . ,A9), be the trouble plot for the chemical reactor trouble. Then its TFDL description, in tabular form, is shown in Figure 7.

4.1.2 Depicting trouble frames

As already remarked, basic trouble frames are guides for classes of troubles. [9] furnishes a template language enriched over that for trouble plots by the foundation of field and phenomena markings. These markings must be matched by trouble plot elements for the trouble frame template to utilize.

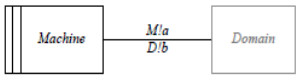
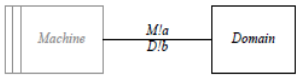
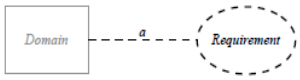
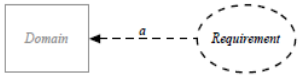
Graphical element	Element	Name	Controlled/ Constrained	Observed/ Referenced
	*M*	*Machine*	*a*	*b*
	*D*	*Domain*	*b*	*a*
	*R*	*Requirement*		*a*
	*R*	*Requirement*	*a*	

Figure 6. The relationship between the graphical components of a trouble plot and their similitude in the TFDL

Each field can be noted as biddable (B), lexical (X) or causal (C):

$$\texttt{PFDomainMarking} \triangleq \texttt{PFDomain} \rightarrow \{\texttt{B}, \texttt{X}, \texttt{C}\}$$

Each (set of) developments can be noticed as causal (C), symbolic (Y) or event (E):

$$\texttt{PFPhenomenaMarking} \triangleq \texttt{PFPhenomena} \rightarrow \{\texttt{C}, \texttt{Y}, \texttt{E}\}$$

The plot of a trouble frame is called a frame plot and is a trouble plot increased with markings for fields and phenomena:

$$\texttt{PFFrameDiagram} \triangleq (\texttt{PFProblemDiagram}, \texttt{PFDomainMarking}, \texttt{PFPhenomenaMarking})$$

A Trouble Frame provides a frame concern, i.e., a template for correctness arguments that is common to all troubles of that class. We assume that frame concerns are taken from another given set:
TFFrameConcern and so we may determine:

$$\texttt{PFProblemFrame} \triangleq (\texttt{PFFrameDiagram}, \texttt{PFFrameConcern})$$

A TFDL delegacy of the controlled conduct frame of Figure 2 is given in Figure 8 (we have omitted the frame concern). The reader will note that the developments assured by the operator are event phenomena (marking E), while all other phenomena are causal (marking C); such markings will constrain the applicability of the trouble frame to a trouble plot.

4.2 Trouble plots as challenges

As in program elaboration [13], in which pre and post circumstances determine a 'challenge' to develop code to enforce them, so a trouble plot, through its surroundings and necessities, finds out a dispute to formulate a machine perticularation to solve them. Whereas refinement is mature, and has formal notation capable of constituting a challenge we, as yet, do not. We will therefore be

	Name	Controlled/Constrained	Observed/Referenced
M	*Operation Machine*	$open\_catalyst_{act}$, $close\_catalyst_{act}$ $increase\_water_{act}$, $decrease\_water_{act}$ *ring_bell*	is_open_{sen}, is_closed_{sen} is_rising_{sen}, $is\_falling_{sen}$ *request_service* *open_catalyst*, *close_catalyst*
D_1	*Catalyst*	is_open_{sen}, is_closed_{sen} *open*, *closed*	$open\_catalyst_{act}$, $close\_catalyst_{act}$
D_2	*Cooling System*	is_rising_{sen}, $is\_falling_{sen}$ *water_level*	$increase\_water_{act}$, $decrease\_water_{act}$
D_3	*Reactor*	$\emptyset$	$\emptyset$
D_4	*Operator*	*open_catalyst*, *close_catalyst*	$\emptyset$
D_5	*GearBox*	*request_service* *oil_level*	$\emptyset$
D_6	*Alarm*	*bell_ringing*	*ring_bell*
R	*Control*	*open*, *closed* *water_level* *bell_ringing*	*open_catalyst*, *close_catalyst* *oil_level*

Figure 7. The Chemical Reactor trouble plot in tabular form

	Name	Domain Marking for	Constrained Phenomena Marking for Controlled\	Referenced Phenomena Marking for Observed\
M	Control Machine	--	C1	C5, E4
Dc	Controlled Domain	C	C5, C3	C1
Dv	Operator	B	E4	C1
R	Commanded Behaviour	--	C3	E4

Figure 8. TFDL description of the required behaviour frame

forced to extemporize, rotationally, somewhat.

We employ the following definition as the basis for our semantics. Given a world verbal description K, a necessity description R, and sets of shared phenomena c (controlled by the machine field) and o (observed by the machine field) over some DRDL, we determine a 'dispute to find a satisfying perticularation' as

$$c, o : [K, R] = \{S \mid S \text{ controls } c \wedge S \text{ observes } o \wedge K, S \vdash_{DRDL} R\}$$

As such, c,o : [K,R] stands for the set of all those pertain that control phenomena in the set c, and celebrate phenomena in the set o, and are capable, in the context of K, of firing R, as per [15][5][6]. As indicated in the notation, the details of `DRDL will be provided by the DRDL. We also note that c, o : [K,R] can be empty (as when R is false, for instance), in which case there is no satisfying perticularation, i.e., the system cannot be built. Concerning to the original Reference Model, c = sv , i.e., the machine controlled phenomena shared with the environment, and o = ev , the machine visible shared developments.

To derive the dispute which stands as semantics for a particular trouble plot, one must only look to its TFDL verbal description. Given a trouble plot ({D1, . . . ,Dn},M,R,A), with A = (A1, . . . ,A9), in the meaning, for convenience we define its challenge as

$$c, o : [DD_1 \wedge ... \wedge DD_n, DR]$$

Where:

- DDi = A2(Di) — the trouble plot's field descriptions;
- DR = A4(R) — the trouble plot's requirement description;
- c = A6(M) — those developments controlled by the trouble plot's machine field;
- And o = A7(M) — those developments visible to the trouble plot's machine field.

The reader will note that firing the correctness argument linked with the plot is part of the challenge: perticularally, demonstrating that K, S `DRDL R, that S controls c, and that
S observes o.

4.3 Instantiation as injection

A trouble frame is a template for a class of troubles. By 'applying' a trouble frame to a trouble plot, we can descend a (sub-) trouble plot whose meaning is a challenge. This sets aside for a procedure of trouble rotting in which the sub-trouble plots are expulsions of the original trouble plot through appropriately chosen trouble frames.

Semantically, we can see the application program of a trouble frame to a trouble plot as corresponding areas and connections (i.e., the topology), and the types of the fields and developments.

We say that a frame plot TF = (D,M,R,A,DM,PM) matches a trouble plot PD = (D0,M0,R0,A0) when there is an injective balance _ : D ! D0 which respects the topology, field and phenomena types. A trouble frame representation is then the application of _ to (D,M, R,A,DM,PM).

4.4 Covering authorship and recomposition

Leading the analogy with program civilization, working towards an S 2 c, o : [K,R] will be an geographic expedition of the design choices at each stage: becaemploy we employment within all of package—including architectures, etc—and not just within code, the design space is maybe much larger than that in refinement.

Trouble frames, as leads, allow us to navigate the designate space by dividing up the trouble. For example, in the Chemical Reactor trouble, we have three sub-trouble plots representing to the required behaviour, commanded behaviour, and information display fleshes. The rotting of this trouble into three sub-troubles is illustrated in Figure 9, in which:

$$\begin{aligned} c_1 &= \{open\_catalyst_{act}, close\_catalyst_{act}\} \\ o_1 &= \{is\_open_{sen}, is\_closed_{sen}, \\ &\quad open\_catalyst, close\_catalyst\} \\ c_2 &= \{increase\_water_{act}, decrease\_water_{act}\} \\ o_2 &= \{is\_raising_{sen}, is\_falling_{sen}\} \\ c_3 &= \{ring\_bell\} \\ o_3 &= \{request\_service\} \\ c &= c_1 \cup c_2 \cup c_3 \\ o &= o_1 \cup o_2 \cup o_3 \end{aligned}$$

As in elaboration, given a possible rotting through the application program of trouble frames, there will be a representing composition concern analogous to refinement's verification condition: we must be able to convince ourselves of the validity of the recomposition. Our point is that the backwards argument (upwards in Figure 9) identifies the sub-troubles that must be

Figured out, while the advocating argument (downwards in Figure 9) requires that printing business is identified and the linked proof obligations are conducted.

In the Chemical Reactor trouble, for instance, there is a necessity that when the oil level in the gearbox is low, the machine should sound an alarm and halt (the information display sub-trouble). On the other hand, the machine is invariably necessitated to regulate the cooling system (the required behaviour sub-trouble). The composition of these two obviously conflicting sub-troubles must continue to guarantee the safe functioning of the reactor. This concern must therefore be covered at the composition level, perhaps by reconsidering one or both of these requirements.

$$\frac{\begin{array}{c} c_1, o_1 : [\{Catalyst, Operator\}, Control\ \ Catalyst] \\ c_2, o_2 : [\{Cooling\ \ System\}, Regulate\ \ Cooling\ \ System] \\ c_3, o_3 : [\{Alarm, GearBox\}, Raise\ \ Alarm] \end{array}}{c, o : [\{Catalyst, Cooling\ \ System, Operator, Alarm, GearBox\}, Control]} \uparrow \text{Decomp.}, \downarrow \text{Recomp.}$$

Figure 9. The rotting/recomposition human relationship among the Chemical reactor trouble plot and its sub trouble plots

The indication that demands are in engagement can be taken from the backward growth of the challenges. The binding of symbols that occurs during the tree through backwards growth may lead to a situation in which the paper concern cannot be fired. This is natural in such proof theoretic models. By analogy, sometimes, when we start from an unobvious 'theorem', it is becaemploy the 'theorem' contains a contradiction and the 'theorem's' statement must be rethought. As with system growth through trouble frames, contravening requirements may prevent the system being built. When we are made aware of such requirements—becaemploy the challenge cannot be met—we must reconsider them, and formulate different, non-conflicting necessities.

V. DISCUSSIONS AND FUTURE WORK

In this research paper we have provided semantics for elements of the TF model. We have introduced the Trouble Frames Description Language (TFDL), over a Field and Requirements Description Language (DRDL), for the description of trouble plots and trouble frames. We have defined the meaning of a trouble frame as a template for trouble plots, and a semantics for trouble plots within the Reference Model, given in terms of challenges of the form c, o : [K,R], a notion that we have also brought in.

We emphasize that our meaning is based on a weak notion of rightness. In this we disagree from the Reference Model semantics, which is based on universal measurement over all 'phenomenon sequences'. The actual wording employed by the authors is:

A requirement with an environment restraint should always be asserted by a demonstration or proof that specified properties, conjoined with field knowledge, guarantee the satisfaction of the necessity.

In our meaning we relax that notion of proof to its weakest (non-technical) meaning: facts, evidence, controversy, etc. establishing or helping to establish a fact.

We might, with this resolution, also admit proofs of the type 'an broad search has found no case in which the system did not satisfy the demands' (a testing 'proof'); or the 'clients were confident by the arguments of the developers'; or 'a formal refinement of the requirements to code was performed, with all verification conditions being discharged'; or 'Dijkstra programmed it'; even 'we'll fix it in the next release' might do. Clearly, the nature of `DRDL and that of the demands are closely related: where the demands include human safety (as may be the case in a critical system), the strength of `DRDL must provide for an appropriate level of proof. The growth of the challenge expression c, o : [K,R] needs many things. Most importantly, the availability of a 'weakest environment predicate transformer' (analogous to the weakest precondition semantics of refinement) would place the semantic ground of the manipulation of such expressions on a much firmer footing, and even lead us towards a requirements engineering notion of 'refinement' that forms a lattice over such objects; given a trouble, the refinement relation v would justify the relationship, summed up in Figure 10:

$$problem \sqsubseteq partial\_soln_0 \sqsubseteq \ldots \sqsubseteq specification$$

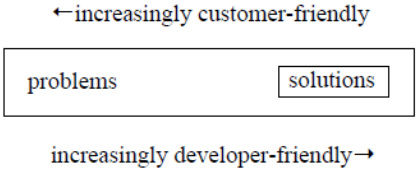


Figure 10. The move among trouble and solution fields (after [13, Figure 1.6, page 9])

Finally, we discover that this research paper provides a basis for a formal semantics for elements of the TF model. The range of employs to which it can be put is still mostly undiscovered, and will furnish a focus for further research.

REFERENCES

[1] D. Bjorner, S. Koussoube, R. Noussi, and G. Satchok. Michael jackson's trouble frames: Towards methodological principles of selecting and applying formal package growth techniques and tools. In 1st IEEE International Conference on Formal Engineering Methods. IEEE Computer Society Press, 1997.

[2] O. Dieste and A. Silva. Requirements: Closing the gap among field and computing knowledge. In Proceedings of SCI2000, Vol. II (Information Systems Growth), 2000.

[3] A. Evans, J.-M. Bruel, R. France, K. Lano, and B. Rumpe. Making UML precise. In L. Andrade, A. Moreira, A. Deshpande, and S. Kent, editors, Proceedings of the OOPSLA'98 Workshop on Formalizing UML. Why? How?, 1998.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns. Addison-Wesley, 1995.

[5] C. Gunter, E. Gunter, M. Jackson, and P. Zave. A reference model for requirements and perticularations. IEEE Package, 3(17):37–43, 2000.

[6] J. G. Hall and L. Rapanotti. A Reference Model for Requirements Engineering. In Proceedings of the 11th International Joint Conference of Requirements Engineering, 2003.

[7] D. Harel and A. Naamad. The STATEMATE semantics of statecharts. ACM Transactions on Package Engineering and Methodology, 5(4):293– 333, 1996.

[8] M. Jackson. Package Requirements & Perticularations: a Lexicon of Practice, Principles, and Prejudices. Addison-Wesley, 1995.

[9] M. Jackson. Trouble Frames. Addison-Wesley, 2001.

[10] S. Konrad and B. H. Cheng. Requirements patterns for embedded systems. In IEEE Joint International Conference on Requirements Engineering, 2002.

[11] P. G. Larsen, N. Plat, and H. Toetenel. A formal semantics of data flow plots. Formal Aspects of Computing, 6(6):586–606, 1994.

[12] N. Leveson. Package safety: why, what and how ACM Computing Survey, 18(2):125–163, 1986.

[13] C. Morgan. Programming from Perticularations. Prentice-Hall International, 1990.

[14] M. Petre. Why looking isn't always seeing: readership skills and graphical programming. Commun. ACM, 38(6):33–44, 1995.

[15] P. Zave and M. Jackson. Four dark corners of requirements engineering. ACM Transactions on Package Engineering and Methodology, 6(1):1–30, January 1997.

