I.J. Information Technology and Computer Science, **2014, 04, 56-65**
Published Online March 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.04.07

A Formal Description of Problem Frames

Souleymane KOUSSOUBE, Roger NOUSSI, Balira O. KONFE
Laboratoire Africain d'Informatique et de Mathématiques Appliquées (LAIMA), Institut Africain d'Informatique (IAI), BP: 2263 Libreville, Gabon
E-mail:skoussoube@gmail.com, roger_noussi@yahoo.fr, obalira@gmail.com

Abstract—Michael Jackson defines a Problem Frame as a mean to describe and classify software development problems. The initial description of problem Frames is essentially graphical. A weakness of this proposal is the lack of formal specification allowing efficient reasoning tools. This paper deals with Problem Frames' formal specification with Description Logics. We first propose a formal terminology of Problem Frames leading to the specification of a Problem Frames' TBOX and a specific problem's ABOX. The Description Logics inference tools can then be used to decompose multi frame problems or to fix a particular problem into a Problem Frame.

Index Terms—Problem Frame, Description Logics, Problem Diagram, Problems Matching, Problem Decomposition

I. Introduction

The concept of Problem Frames (PF) has been introduced by Michael Jackson [1, 2 and 3] as a mean for describing and classifying problems. These problem classes can then be related to appropriate tools or methods. PF description includes:

- the world in which the problem is located;
- the machine to build and;
- the requirement. It is the condition in the problem domain that the machine must guarantee.

Let us note that PF describes problems, instead of their solutions. The description of PF is given in a graphical way. This has fundamental disadvantages [4]; for example:

- Some misunderstandings can easily occur when we interpret graphical artifacts;
- It is not easy to verify the completeness and the correctness of the description.
- It is not easy to identify equivalent structures that could be used interchangeably.

The first attempt of formal characterization of PF was done in [5].The semantic of PF is clarified in [6] but the languages used still have a lack of formality. Other works [7 and 8] propose a formal description of PF using ontology. However these proposals don't define efficient formal reasoning tools on PF. **This paper deals with a formal specification of PF using Description Logics (DL)**. DL formalism allows us to get advantage of an environment with a clear and precise syntax and semantics. Furthermore it gives way to use of DL inference tools to match specific problems with PF. It also facilitates the use of formal operations (transformation, decomposition, integration etc.) on PF. The paper is organized as follows: In section II, we present the PF framework. The section III is devoted to a brief presentation of DL with emphasis on their semantics and inference mechanisms. The section IV deals with the PF formalization, including the PF TBOX, the problem diagram ABOX, and the reasoning tools on PF and particular problems. A problem decomposition example is given in the section V.

II. Problem Frame Framework

In this section we review some basic elements of the PF framework.

2.1 Problem Frame

The software development task [9] is to design and construct an artifact. In PF framework [1 and 2], this artifact is called the *machine*, constructed by building a software. The machine is used to meet a recognized need, which is called the *requirement*. Satisfying the requirement involves transforming the physical world around. The component of the world in which the requirement is located and that must be transformed, is called the *problem world*. So, the principal components of a software development problem are: the machine, the problem world, and the requirement. Their relationships are shown in the generalized PF diagram in Fig.1 below.

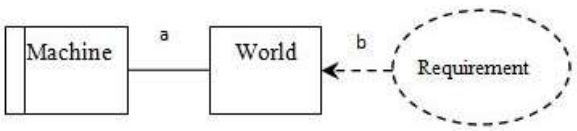


Fig. 1: General Software engineering Problem

The machine interacts with the problem world at an interface of shared phenomena *a*. Typically, these

phenomena are *events, operations* or *states*, controlled either by the problem world or by the machine. The requirement is shown by a dashed oval, indicating its intangible quality. The requirement is not a tangible part of the problem: it is a predicate or condition on the problem world that the machine must guarantee.

One of the aims of the PF framework is to identify basic classes of problem that recur throughout software development. A problem frame acts as a template for recognizing a problem in its class. A particular PF elaborates and specializes the general form of Fig.1 in the following ways:

- The World is decomposed into domains referred to, as **world domain**, in the remainder of this paper.
- Different types of domains are distinguished according to their role in the problem (B for biddable, C for causal…).
- Interfaces of phenomena shared between domains are shown.
- The connections among the domains are more closely characterized in terms of the types of connecting phenomena (events, states, operations).
- The phenomena related by the requirements are similarly characterized according to their types.
- The characteristics of domains interfaces are classified.

For example, the commanded behavior frame [2] can be described by a graphical notation as follows:

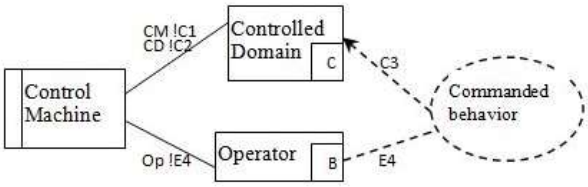


Fig. 2: The commanded behavior frame

- The world is decomposed into two domains : The controlled domain and the operator domain.
- The controlled domain is a causal domain while the operator is a biddable domain.
- The controlled domain and the control machine share the causal phenomena C1 and C2. The control machine and the operator share the event E4. The requierement is a predicate with arguments E4 and C3. C1, C2 and C3 are states and E4 is an event.
- The phenomenon C1 is controlled by the Control Machine, C2 by the controlled domain and E4 by the operator.

2.2 Problem Diagram

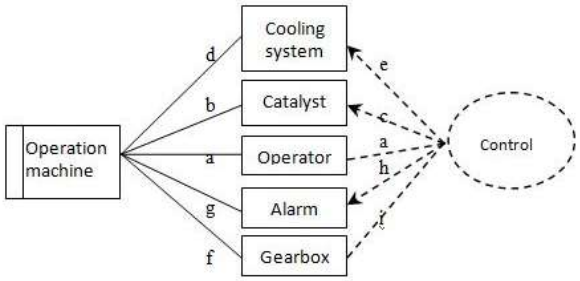
Within the PF framework, a problem diagram defines the 'shape' of a specific problem. It captures the characteristics and interconnections of the components of the world it is concerned with. A problem diagram also includes the requirements that constrain the relationships between these components. In the graphical notation, a problem diagram uses the same symbols as the PF.

To focus our review, we present the following example on a Chemical Reactor Controller described in [2]:

A computer system is required to control the catalyst unit and cooling system of a chemical reactor. An operator issues commands for activating or deactivating the catalyst unit; in response to such commands, the system instructs the unit accordingly and regulates the flow of cooling water. A gearbox is attached to the system: whenever the oil level in the gearbox is low, the system should ring a bell and halt execution.

The problem diagram for the Chemical Reactor Controller problem is shown in Fig.3. The components are:

- A double-barred box (Operation Machine): the machine domain, i.e., the software system to build with its underlying hardware.
- A box (the Operator domain): the human operator. Human operators are regarded as biddable in the PF framework: they may obey stipulated procedures, but not reliably, and may generate events spontaneously.
- Other boxes (Cooling System, Catalyst, etc.): given domains representing components of the world. These domains are causal: i.e., its phenomena are physical events and states, and are causally related;



a : OP!{OpenCatalyst, CloseCatalyst}

b : OM!{$OpenCatalyst_{act}$, $CloseCatalyst_{act}$} CA!{$IsOpen_{sen}$, $IsClosed_{sen}$}

c : CA!{Open, Closed}

d : OM!{$IncreaseWater_{act}$, $DecreaseWater_{act}$} CS!{$IsRising_{sen}$, $IsFalling_{sen}$}

e : CA!WaterLevel

f : GB!RequestService

g : OM!RingBell

h : AL!BellRinging

i : GB!OilLevel

Fig. 3: The Chemical Reactor Problem Diagram

III. Description Logics and Knowledge-Based Systems

Description Logics (DL) [10 and 11] are knowledge representation formalisms used to describe concepts in a given domain. A knowledge base (**KB**) described in DL has two components, the TBOX and the ABOX. The TBOX introduces the terminology, i.e., the vocabulary of an application domain, while the ABOX contains assertions about named individuals in terms of this vocabulary. The vocabulary consists of concepts, denoting sets of individuals (identified objects of the domain), and roles (binary relationships between individuals). In addition to atomic concepts and roles, all DL systems allow building complex descriptions of concepts and roles. Depending on provided operators, there are several DL languages, the **Attributive Language (*AL*)** being the minimal one. We summarize here the syntax and the semantics of some DL languages.

3.1 Syntax and Semantics of DL Languages

Concepts and roles are inductively defined from a set $\boldsymbol{N_C}$ of concepts names (atomic concepts), a set $\boldsymbol{N_R}$ of roles names (atomic roles) and a set of operators.

In the following, unless otherwise stated, A and B are elements of $\boldsymbol{N}_C$; *r* and *s* are components of $\boldsymbol{N}_R$;

C and *D* are concepts descriptions and *n* is a positive integer.

The minimal language ***AL*** contains the atomic concepts, the universal concept, the bottom concept, atomic negation, intersection, value restriction and limited existential quantification.

$$AL = \{A, \mathrm{T}, \bot, \neg A, C \cap D, \forall r \bullet C, \exists r \bullet T\}$$

Extending ***AL*** by any subset of the constructors in the table below, yields a particular ***AL***-language. Each ***AL***-language is denoted by a string of the form:

$$AL[U][\varepsilon][N][C][Q][I]$$

Hence, *ALCQI* is the language obtained from AL by adding full negation(*C*), qualified number restriction (*Q*) and Inverse of role (*I*).

In order to define a formal semantics of concepts descriptions, we consider an interpretation *I* that consists of a non-empty set Δ^I (the domain of the interpretation) and an interpretation function $\bullet^I$, which assigns to every atomic concept *A*, a set $A^I \subseteq \Delta^I$ and to every atomic role *r* a binary relation $r^I \subseteq \Delta^I \times \Delta^I$.

For example, we have the following interpretations:

- T is the whole domain, i.e. all the individuals in the domain.
- $\forall r \bullet C$ is the set of individuals who are related through *r* only to individuals satisfying C.
- $\exists r \bullet T$ is the set of individuals related through *r* with other individuals of the domain.
- $\leq nr \bullet C$ is the set of individuals who are related through *r* to at most n individuals satisfying *C*.

The following table summarizes the syntax and the semantics of DL.

3.2 Inference Techniques in Description Logics

At the terminological level, there are four inference operators:

- **Satisfiability**: A concept *C* of a terminology *T* is satisfiable if and only if there exists a model *I* of *T* such that $C^I \neq \{\ \}$
- **Subsumption**: A concept *C* is subsumed by a concept D ($C \sqsubseteq D$) for a terminology *T* if and only if $C^I \subseteq D^I$ for any model *I* of *T*.
- **Equivalence**: A concept *C* is equivalent to a concept *D* ($C \equiv D$) for a terminology *T* if and only if $C^I = D^I$ for each model *I* of *T*.
- **Disjunction**: Concepts *C* and *D* are disjoined by report/ratio terminology *T* if and only if $C^I \setminus D^I = \{\ \}$; for each model *I* of *T*.

The 4 types of problems can be brought back to problems of subsumption or satisfiability. Consequently, the design of inference engines requires, very often only one type of algorithm.

At the factual level, there are also four inference operators:

- **Coherence**: An ABOX A is coherent with reference to a TBOX *T* if and only if there exists a model I of *A* and *T*.
- **Checking of authority**: To check by inference if an assertion *C(a)* is true for any model *I* of an ABOX *A* and a TBOX *T*.
- **Checking of role**: To check by inference if an assertion *r(a; b)* is true for any model *I* of an ABOX *A* and a TBOX *T* .
- **Recovery problem**: For an ABOX *A*, a concept *C* of a terminology *T* , infer the individuals a such that *C(a)*

One can find non standard inference operators in [12].

Table 1: Syntax and semantic of DL

Syntax	Definition	Semantic	Symbol
$\top$	Universal concept	$\top^I = \Delta^I$	*AL*
$\bot$	Bottom concept	$\bot^I = \emptyset$	*AL*
$\neg A$	Atomic negation	$(\neg A)^I = \Delta^I \setminus A^I$	*AL*
C П D	intersection	$(C \cap D)^I = C^I \cap D^I$	*AL*
$\forall r \bullet C$	Value restriction	$(\forall r \bullet C)^I = \{a \in \Delta^I \mid \forall b \bullet (a,b) \in r^I \Rightarrow b \in C^I \}$	*AL*
$\exists r \bullet \top$ or $\exists r$	Limited existential quantification	$(\exists r \bullet \top)^I = \{a \in \Delta^I \mid \forall b \bullet (a,b) \in r^I \}$	*AL*
$\exists r \bullet C$	Full existential quantification	$(\exists r \bullet C)^I = \{a \in \Delta^I \mid \forall b \bullet (a,b) \in r^I \wedge b \in C^I \}$	ε
C ⊔D	union	$(C \cup D)^I = C^I \cup D^I$	*U*
$\neg C$	Full negation	$(\neg C)^I = \Delta^I \setminus C^I$	*C*
$\leq nr \bullet C$	At most qualified number restriction	$(\leq nr \bullet C)^I = \{a \in \Delta^I \mid \{b \mid (a,b) \in r^I \} \leq n \wedge b \in C^I \}$	*Q*
$\geq nr \bullet C$	At least qualified number restriction	$(\geq nr \bullet C)^I = \{a \in \Delta^I \mid \{b \mid (a,b) \in r^I \} \geq n \wedge b \in C^I \}$	*Q*
$\leq nr$	At most unqualified number restriction	$(\leq nr)^I = \{a \in \Delta^I \mid \{b \mid (a,b) \in r^I \} \leq n \}$	*N*
$\geq nr$	At least unqualified number restriction	$(\geq nr)^I = \{a \in \Delta^I \mid \{b \mid (a,b) \in r^I \} \geq n \}$	*N*
r^-	Inverse role	$(r^-)^I = \{(b,a) \in \Delta^I \times \Delta^I \mid (a,b) \in r^I \}$	*I*

IV. Description Logics Formalism for Problem Frames

In this section, we propose an approach to obtain a complete formal specification of PF using Description Logics. First, we fix the formal terminology of the PF framework. Using this terminology, we build the formal specification of a PF and the formal description of a specific problem diagram. Finally, we introduce the reasoning tools.

4.1 The Settings

We begin with a UML-like specification of the PF domain which points out the main concepts with their relationships.

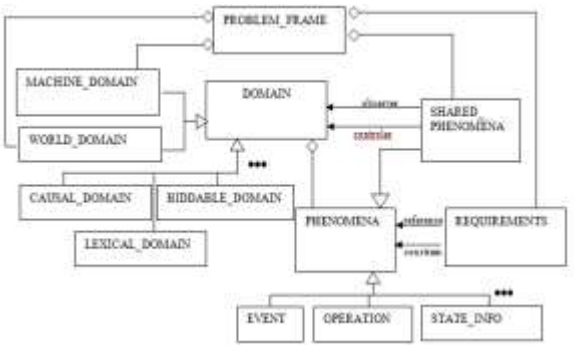


Fig. 4: UML-Like Representation of the PF domain

In Fig.4, it appears that a PF aggregates a machine domain, a set of world domains, a set of shared phenomena and the requirements. From this diagram,

we produce the following PF formal Terminology in the form of a TBOX. Translation rules of a UML class diagram into a TBOX can be found in [13]. The concepts of the PF terminology are given in table 2 below.

Table 2: the concepts of the PF Terminology

1. DOMAIN, PROBLEM, PROBLEMFRAME
2. MACHINE_DOMAIN⊑ DOMAIN Π Observer•SHAREDPHENOMENA
3. WORD_DOMAIN DOMAIN Π MACHINE_DOMAIN,
4. REQUIREMENT ⊑ constraint•PHENOMENA ⊔ reference•PHENOMENA,
5. PHENOMENA ⊑ belong•DOMAIN ,
6. SHAREDPHENOMENA PHENOMENA Π controller•DOMAIN Π *1.controller•DOMAIN* Π *2 Observer•DOMAIN*,
7. CAUSAL_DOMAIN ⊑ WORLD_DOMAIN,
8. BIDDABLE_DOMAIN ⊑ WORLD_DOMAIN,
9. LEXICAL_DOMAIN ⊑ WORLD_DOMAIN,
10. SYMBOLIC_DOMAIN ⊑ WORLD_DOMAIN,
11. EVENT ⊑ PHENOMENA,
12. OPERATION ⊑ PHENOMENA,
13. STATE_INFORMATION ⊑ PHENOMENA
14. CAUSAL_ PHENOMENA ⊑ PHENOMENA
15. PROBLEMFRAME⊑ IncludeDom•MACHINE_DOMAIN Π IncludeDom•WORLD_DOMAIN Π IncludeShPhe• SHAREDPHENOMENA Π IncludeReq• REQUIREMENT

In the Tab.2, statements 1 to 6 specify the main components of the PF, namely, domain, machine, world, phenomena, shared phenomena and requirement. Statement 2 is a terminological axiom specifying that the machine domain is a domain and is the observer of some shared phenomena. This axiom is a minimal constraint that can be refined into a definition. Statement 3 specifies that the machine domain is not a component of the world domain. Statement 4 indicates that a requirement constraints or references a phenomenon (in a domain). Statement 6 indicates that shared phenomena are phenomena. They must be controlled by one domain and observed by at least two domains. Statements 7 to 13 specify taxonomic relations between these concepts and others; they also use the roles (observer, relates, belong…) presented in the Tab.3 below.

Table 3: the roles of the PF Terminology

Roles	Related Concepts
includeDom	ProblemFrame, Domain
includeShPhe	ProblemFrame, Phenomena
includeReq	ProblemFrame, Requirement
Belongs	Phenomena, Domain
controller	SharedPhenomena, Domain
Observer	SharedPhenomena, Domain
Constraint	Requirement, Phenomena
Reference	Requirement, Phenomena
relates Reference ⊔ Constraint	
controller⊑ observer	

4.2 The Problem Frame TBOX

In our approach, a PF is specified by a TBOX using the PF terminology. The TBOX describes the components of the PF. In this description, a top level concept represents the described PF. The formal description in DL language of the commanded behavior frame (see Fig.2) can be given as follows:

Table 4: the TBox of the Commanded Behavior Frame

CBehaviorFrame CBehaviorFrameDomain Π CBehaviorFrameSharedPhen Π CBehaviorFrameRequirement
CBehaviorFrameDomain includeDom•MACHINE_DOMAIN Π includeDom•CAUSAL_DOMAIN Π includeDom•BIDDABLE_DOMAIN
CBehaviorFrameReq includeReq•(REQUIREMENT Π references• (EVENT Π belongs•BIDDABLE_DOMAIN) Π constraint•(CAUSAL_ PHENOMENA Π belongs•CAUSAL_DOMAIN))
CBehaviorFrameSharedPhen includeSharedPhen• (CAUSAL_ PHENOMENA Π observer•CAUSAL_DOMAIN Π controller•MACHINE_DOMAIN) Π includeSharedPhen• (CAUSAL_ PHENOMENA Π observer•MACHINE_DOMAIN Π controller• CAUSAL_DOMAIN) Π includeSharedPhen• (EVENT Π observer• MACHINE_DOMAIN Π controller•BIDDABLE_DOMAIN

The top level concept *CBehaviorFrame* represents the commanded behavior frame. This concept must obviously satisfy the terminological axiom given in item15 of the PF terminology i.e*: CBehaviorFrame* ⊑ *PROBLEMFRAME*

CBehaviorFrameDomain, *CBehaviorFrameReq* and *CBehaviorFrameSharedPhen* specify respectively the domains, the requirements and the shared phenomena of the commanded behavior frame.

4.3 The Problem Diagram ABOX

We encode the problem diagram is an ABOX which records assertions of the diagram. The domains, including the machine, the phenomena, as well as the requirement are represented as named individuals. A special individual representing the specific problem is introduced as an instance of the concept PROBLEM.

The assertions of the ABOX have two forms:

- C(a) where *C* is a concept of the PF terminology defined in section 4.1 and *a* is a named individual, states that *a* is an instance of *C*.

- r(a,b) where *r* is a role of the PF terminology and *(a,b)* a pair of individuals states that there exists a relation *r* between *a* and *b*.

The ABOX of the chemical reactor problem is given in Tab.5a and Tab.5b below. In the given description, a special individual *ChemicalReactorProblem* represents the problem. The Tab.5a introduces the named individuals and the concepts to which they belong. For example, item 4 specifies that there is an individual *catalyst* which is an instance of the concept *CAUSAL_DOMAIN*; item 14 specifies that there is an individual *OpenCatalyst* which is an instance of the concept *EVENT*. Item 19 specifies that there is an individual $OpenCatalyst_{act}$ which is an instance of the concept *SHAREDPHENOMENA*. Item 08 specifies that there is an individual *Control* which is a *REQUIREMENT*.

Table 5a: The Chemical Reactor ABOX (part1)

1 - PROBLEM (ChemicalReactorProblem)
2 – MACHINE_DOMAIN(OperationMachine)
3 – CAUSAL_DOMAIN(CoolingSystem)
4 – CAUSAL_DOMAIN(Catalyst)
5 – CAUSAL_DOMAIN(Alarm)
6 – CAUSAL_DOMAIN(GearBox)
7 – BIDDABLE_DOMAIN(Operator)
8 - REQUIREMENT(Control)
9 - CAUSAL_PHENOMENA(WaterLevel)
10 - CAUSAL_PHENOMENA(Open)
11 - CAUSAL_PHENOMENA(Closed)
12 - CAUSAL_PHENOMENA(BellRinging)
13 - EVENT (CloseCatalyst)
14 - EVENT (OpenCatalyst)
15 - CAUSAL_PHENOMENA(OilLevel)
16 - SHAREDPHENOMENA(RingBell)
17 - SHAREDPHENOMENA($IncreaseWater_{act}$)
18 - SHAREDPHENOMENA($DecreaseWater_{act}$)
19 - SHAREDPHENOMENA($OpenCatalyst_{act}$)
19b - SHAREDPHENOMENA($CloseCatalyst_{act}$)
20 - SHAREDPHENOMENA(CloseCatalyst)
20b - SHAREDPHENOMENA(OpenCatalyst)
21 - SHAREDPHENOMENA($IsOpen_{sen}$)
22 - SHAREDPHENOMENA($IsClose_{sen}$)
23 - SHAREDPHENOMENA($IsRising_{sen}$)
24 - SHAREDPHENOMENA($IsFalling_{sen}$)

Tab.5b specifies relations between the individuals; for example, item 27 specifies that there is a relation *IncludeDom* between the individuals *ChemicalReactorPb* and *Catalyst*. This relation means that the causal domain *Catalyst* is a domain of the chemical reactor problem. Item 35 specifies a relation *IncludeShPhe* between the individuals *ChemicalReactorPb* and $OpenCatalyst_{act.}$ This relation means that $OpenCatalyst_{act}$ is a shared phenomenon in the chemical reactor problem.

Table 5b: The Chemical Reactor ABOX (part2)

25 - IncludeDom(ChemicalReactorPb,OperationMachine)
26 - IncludeDom(ChemicalReactorPb,CoolingSystem)
27 - IncludeDom(ChemicalReactorPb, Catalyst)
28 - IncludeDom(ChemicalReactorPb, Alarm)
29 - IncludeDom(ChemicalReactorPb, GearBox)
30 - IncludeDom(ChemicalReactorPb, Operator)
31 - IncludeReq(ChemicalReactorPb, Control)
32 - IncludeShPhe(ChemicalReactorPb, RingBell)
33 - IncludeShPhe(ChemicalReactorPb, $IncreaseWater_{act}$)
34 - IncludeShPhe(ChemicalReactorPb, $DecreaseWater_{act}$)
35 - IncludeShPhe(ChemicalReactorFrame, $OpenCatalyst_{act}$)
35b - IncludeShPhe(ChemicalReactorFrame, OpenCatalyst)
36 - IncludeShPhe(ChemicalReactorPb, $CloseCatalyst_{act}$)
36b - IncludeShPhe(ChemicalReactorPb, CloseCatalyst)
37 - IncludeShPhe(ChemicalReactorPb, $IsOpen_{sen}$)
38 - IncludeShPhe(ChemicalReactorPb, $IsClose_{sen}$)
39 - IncludeShPhe(ChemicalReactorPb, $IsRising_{sen}$)
39b - IncludeShPhe(ChemicalReactorPb, RequestService)
40 - IncludeShPhe(ChemicalReactorPb, $IsFalling_{sen}$)
41 - Constraint(Control, WaterLevel)
42 - Constraint(Control, Open)
43 - Constraint(Control, Close)
44 - Constraint(Control, BellRinging)
45 - Reference(Control, OpenCatalyst)
46 - Reference(Control, CloseCatalyst)
47 - Reference(Control, OilLevel)
48 - Belongs(WaterLevel, CoolingSystem)
49 - Belongs(Open, Catalyst)
50 - Belongs(Close, Catalyst
51 - Belongs(OpenCatalyst, Operator)
52 - Belongs(CloseCatalyst, Operator)
53 - Belongs(BellRinging, Alarm)
54 - Belongs(OilLevel, GearBox)
55 - Controler(OpenCatalyst, Operator)
56 - Controler(CloseCatalyst, Operator)
57 - Controler($OpenCatalyst_{act}$, OperationMachine)
58 - Controler($CloseCatalyst_{act}$, OperatorMachine)
59 - Controler($IsOpen_{sen}$, Catalyst)
60 - Controler($IsClose_{sen}$, Catalyst)
61 - Controler($IncreaseWater_{act}$, OperationMachine)
62 - Controler($DecreaseWater_{act}$, OperationMachine)
63 - Controler($IsRising_{sen}$, Cooling_System)
64 - Controler($IsFalling_{sen}$, CoolingSystem)
65 - Controler(RingBell, OperationMachine)
66 - Controler(RequestService, GearBox)
67 - Observer(OpenCatalyst, OperationMachine)
68 - Observer(CloseCatalyst, OperationMachine)
69 - Observer($OpenCatalyst_{act}$, Catalyst)
70 - Observer($CloseCatalyst_{act}$, Catalyst)
71 - Observer($IsOpen_{sen}$, OperationMachine)
72 - Observer($IsClose_{sen}$, OperationMachine)
73 - Observer($IncreaseWater_{act}$, CoolingSystem)
74 - Observer($DecreaseWater_{act}$, CoolingSystem)
75 - Observer(IsRisingsen, OperationMachine)
76 - Observer(isFallingsen, OperationMachine)
77 - Observer(RingBell, Alarm)
78 - Observer(RequestService, OperationMachine)

Item 51 specifies a relation *Belongs* between the individuals *OpenCatalyst*, and *Operator*. This means that the phenomenon *OpenCatalyst* belongs to the domain *Operator*. Items 55 and 67 mean that phenomenon *OpenCatalyst* is controlled by the domain *operator* and observed by the domain *OperationMachine.* Item 41 specifies that the requirement *(Control)* constraints the phenomenon

WaterLevel belonging to the domain *CoolingSystem* (item 48). The requirement references the phenomenon *OpenCatalyst* in the *Operator* domain (item 45 and item 51).

4.4 Reasoning with PFs

In the previous section, we proposed tools to represent PFs and problems diagrams in DL formalism. This section is devoted to the inference tools. Indeed, we show how to match a problem diagram and a PF.

To match some problem into a PF template, we need to check the following elements:

- the PF topology;
- the domains characteristics;
- the shared phenomena and ;
- the requirements.

In our formal framework, we proceed as follows:

Given:

- a PF TBOX with $\boldsymbol{C^{PF}}$ the top level concept representing the PF;
- a problem ABOX with $\boldsymbol{I^{PB}}$ the individual representing the problem

Assertion 1*: the problem* $\boldsymbol{I^{PB}}$ *maps the Problem Frame* C^{PF} *if* $\boldsymbol{C^{PF}(I^{PB})}$**.** ie the individual $\boldsymbol{I^{PB}}$ is an instance of the concept C^{PF} .

This is done by the basic authority checking operation of the DL engine.

Generally, a problem can map more than one PF. Given a set of PF concepts $\{\boldsymbol{C}_i^{PF}\}$ the previous assertion is then rewritten as follows:

Assertion 2*: the problem* $\boldsymbol{I^{PB}}$ *maps the Problem Frame family* $\{\boldsymbol{C}_i^{PF}\}$ *if:* $\boldsymbol{C}_i^{P}$ $(\boldsymbol{I^{PB}})$, for each i

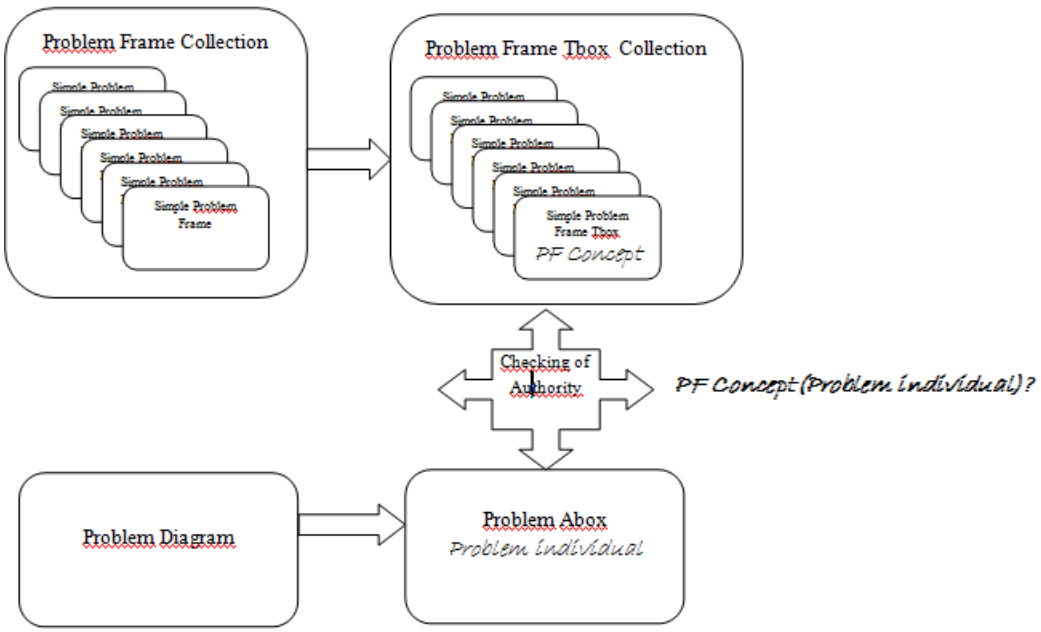


Fig. 5: Problems and PF matching model

The basic instance checking operation can be used as part of a composite inference task involving several PF TBOX as shown in Fig.5 above. This composite instantiation offers the basis for the problem decomposition task.

V. Example of Problem Decomposition

In this section, we propose the decomposition of the chemical reactor problem into the *required behavior frame,* the *information display frame* and the *commanded behavior frame*.

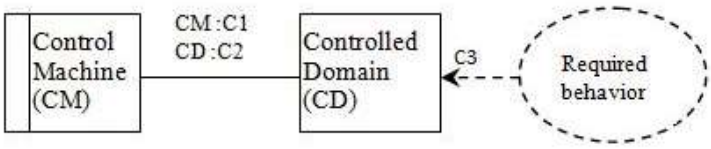


Fig. 6: the required behavior Frame

Table 6: matching the required behavior TBOX with the chemical reactor ABOX

The required behavior frame TBOX
ReqBehaviorFrame ReqBehaviorDomainΠ ReqBehaviorSharedPhenΠ ReqBehaviorRequirement ReqBehaviorDomain includeDom•MACHINE_DOMAIN Π includeDom•CAUSAL_DOMAIN Π ReqBehaviorReq includeReq•(REQUIREMENT Π Π constraint•(CAUSAL_PHENOMENAΠ belongs•CAUSAL_DOMAIN)) ReqBehaviorSharedPhen includeSharedPhen•(CAUSAL_PHENOMENA Π controller•MACHINE_DOMAIN Π observer•CAUSAL_DOMAIN) Π includeSharedPhen•(CAUSAL_PHENOMENA Π observer•MACHINE_DOMAIN Π controller•CAUSAL_DOMAIN
The required behavior instance of the chemical reactor ABOX
1 - PROBLEM (ChemicalReactorProblem) 25 - IncludeDom(ChemicalReactorPb,OperationMachine) 26 - IncludeDom(ChemicalReactorPb,CoolingSystem) 2 – MACHINE_DOMAIN(OperationMachine) 3 – CAUSAL_DOMAIN(CoolingSystem) 31 - IncludeReq(ChemicalReactorPb, Control) 8 - REQUIREMENT(Control) 9 – CAUSAL_PHENOMENA(WaterLevel) 48 - Belongs(WaterLevel, CoolingSystem) 41 - Constraint(Control, WaterLevel) 33 - IncludeShPhe(ChemicalReactorPb, IncreaseWater$_{act}$) 17 - SHAREDPHENOMENA(IncreaseWater$_{act}$) 61 - Controler(IncreaseWater$_{act}$, OperationMachine) 73 - Observer(IncreaseWater$_{act}$, CoolingSystem) 34 - IncludeShPhe(ChemicalReactorPb, DecreaseWater$_{act}$) 18 - SHAREDPHENOMENA(DecreaseWater$_{act}$) 62 - Controler(DecreaseWater$_{act}$, OperationMachine) 74 - Observer(DecreaseWater$_{act}$, CoolingSystem) 39 - IncludeShPhe(ChemicalReactorPb, IsRising$_{sen}$) 23 - SHAREDPHENOMENA(IsRising$_{sen}$) 63 - Controler(IsRising$_{sen}$, CoolingSystem) 75 - Observer(IsRisingsen, OperationMachine) 40 - IncludeShPhe(ChemicalReactorPb, IsFalling$_{sen}$ 24 - SHAREDPHENOMENA(IsFalling$_{sen}$) 64 - Controler(IsFalling$_{sen}$, CoolingSystem) 76 - Observer(IsFallingsen, OperationMachine)

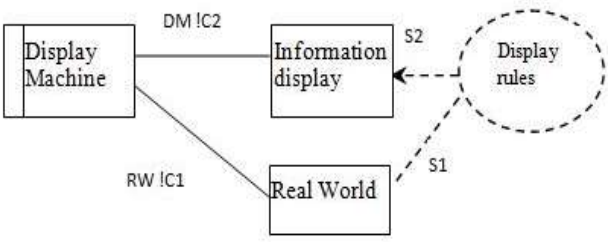


Fig. 7: the information display Frame

Table 7: matching the information display TBOX with the chemical reactor ABOX

The information display frame TBOX
InformationDispFrame informationDispDomainΠ informationDispSharedPhenΠ informationDispRequirement InformationDispDomain includeDom•MACHINE_DOMAIN Π *2* includeDom•CAUSAL_DOMAIN Π InformationDispReq includeReq•(REQUIREMENT Π Π constraint•(CAUSAL_PHENOMENAΠ belongs•CAUSAL_DOMAIN)) Π includeReq•(REQUIREMENT Π reference(CAUSAL_PHENOMENAΠ belongs•CAUSAL_DOMAIN)) InformationDispSharedPhen includeSharedPhen•(CAUSAL_PHENOMENA Π controller•MACHINE_DOMAIN Π observer•CAUSAL_DOMAIN) Π includeSharedPhen•(CAUSAL_PHENOMENA Π controller• CAUSAL_DOMAIN Π observer• MACHINE_DOMAIN)
The information display instance of the chemical reactor ABOX
1 - PROBLEM (ChemicalReactorProblem) 25 - IncludeDom(ChemicalReactorPb,OperationMachine) 28 - IncludeDom(ChemicalReactorPb, Alarm) 29 - IncludeDom(ChemicalReactorPb, GearBox) 2 – MACHINE_DOMAIN(OperationMachine) 5 – CAUSAL_DOMAIN(Alarm) 6 – CAUSAL_DOMAIN(GearBox) 31 - IncludeReq(ChemicalReactorPb, Control) 8 - REQUIREMENT(Control) 12 - CAUSAL_PHENOMENA(BellRinging) 44 - Constraint(Control, BellRinging) 53 - Belongs(BellRinging, Alarm) 15 - CAUSAL_PHENOMENA(Oil_level) 54 - Belongs(OilLevel, GearBox 47 - Reference(Control, OilLevel) 32 - IncludeShPhe(ChemicalReactorPb, RingBell) 16 - SHAREDPHENOMENA(RingBell) 65 - Controler(RingBell, OperationMachine) 77 - Observer(RingBell, Alarm) 39b - IncludeShPhe(ChemicalReactorPb, RequestService) 66 - Controler(RequestService, GearBox) 78 - Observer(equestService, OperationMachine)

Table 8: matching the commanded behavior frame TBOX with the chemical reactor ABOX

The commanded behavior frame TBOX
CBehaviorFrame CBehaviourFrameDomainΠ CBehaviourFrameSharedPhenΠ CBehaviourFrameRequirement CBehaviorFrameDomain includeDom•MACHINE_DOMAIN Π includeDom•CAUSAL_DOMAIN Π includeDom•BIDDABLE_DOMAIN

CBehaviourFrameReq includeReq•(REQUIREMENT Π references• (EVENT Π belongs•BIDDABLE_DOMAIN) Π constraint•(CAUSAL_ PHENOMENA Π belongs•CAUSAL_DOMAIN)) CBehaviourFrameSharedPhen includeSharedPhen• (CAUSAL_ PHENOMENA Π observer•CAUSAL_DOMAIN Π controller•MACHINE_DOMAIN) Π includeSharedPhen• (CAUSAL_ PHENOMENAΠ observer•MACHINE_DOMAIN Π controller• CAUSAL_DOMAIN) Π includeSharedPhen• (EVENT Π observer• MACHINE_DOMAIN Π controller•BIDDABLE_DOMAIN

The commanded behavior instance of the chemical reactor ABOX

1 - PROBLEM (ChemicalReactorProblem)

25 - IncludeDom(ChemicalReactorPb,OperationMachine)
27 - IncludeDom(ChemicalReactorPb, Catalyst)
30 - IncludeDom(ChemicalReactorPb, Operator)
2 – MACHINE_DOMAIN(OperationMachine)
4 – CAUSAL_DOMAIN(Catalyst)
7 – BIDDABLE_DOMAIN(Operator)

31 - IncludeReq(ChemicalReactorPb, Control)
8 - REQUIREMENT(Control)
13 - EVENT(CloseCatalyst)
14 - EVENT (OpenCatalyst)
45 - Reference(Control, OpenCatalyst)
46 - Reference(Control, CloseCatalyst)
51 - Belongs(OpenCatalyst, Operator)
52 - Belongs(CloseCatalyst, Operator)
10 - CAUSAL_PHENOMENA(Open)
11 - CAUSAL_PHENOMENA(Closed)
42 - Constraint(Control, Open)
43 - Constraint(Control, Close)
49 - Belongs(Open, Catalyst)
50 - Belongs(Close, Catalyst

35 - IncludeShPhe(ChemicalReactorFrame, $OpenCatalyst_{act}$)
19 - SHAREDPHENOMENA($OpenCatalyst_{act}$)
57 - Controler($OpenCatalyst_{act}$, OperationMachine)
69 - Observer($OpenCatalyst_{act,}$ Catalyst)
36 - IncludeShPhe(ChemicalReactorPb, $CloseCatalyst_{act}$)
19b - SHAREDPHENOMENA($CloseCatalyst_{act}$)
58 - Controler($CloseCatalyst_{act}$, OperationMachine)
70 - Observer($CloseCatalyst_{act}$, Catalyst)
35b - IncludeShPhe(ChemicalReactorFrame, OpenCatalyst)
14 - EVENT (OpenCatalyst)
55 - Controler(OpenCatalyst, Operator)
67 - Observer(OpenCatalyst, OperationMachine)
36b - IncludeShPhe(ChemicalReactorPb, CloseCatalyst)
13 - EVENT (CloseCatalyst)
56 - Controler(CloseCatalyst, Operator)
68 - Observer(CloseCatalyst, OperationMachine)
38 - IncludeShPhe(ChemicalReactorPb, $IsClose_{sen}$)
22 - SHAREDPHENOMENA($IsClose_{sen}$)
60 - Controler($IsClose_{sen}$, Catalyst)
72 - Observer($IsClose_{sen}$, OperationMachine)
37 - IncludeShPhe(ChemicalReactorPb, $IsOpen_{sen}$)
21 - SHAREDPHENOMENA($IsOpen_{sen}$)
59 - Controler($IsOpen_{sen}$, Catalyst)
71 - Observer($IsOpen_{sen}$, OperationMachine)

We use the chemical reactor ABOX presented in section 4.3, Tab.5a and Tab.5b. For each frame, we present the frame diagram, the corresponding TBOX and the items of the chemical reactor ABOX used by the DL inference engine to check the assertion 1, $\boldsymbol{C^{PF}(I^{PB})}$ where $\boldsymbol{I^{PB}}$ = *ChemicalReactorPb* and C^{PF} is the concept for each frame.

On the left of each item, the number corresponds to the position of this item in Chemical reactor ABOX.

The result of the matching operation has three components according to the TBOX specification. The first section instantiates the domain. The second one instantiates the requirement. The third part concerns the shared phenomena.

Each item of the chemical reactor ABOX appears in at least one of the three instances of the commanded behavior, required behavior or the information display frame. The composition of these three frames, build the frame of the chemical reactor problem.

VI. Conclusion

In this paper, the main topic is to propose a formal approach for the specification of PF. Our proposal is the use of DL. DL brings about a clear syntax, a precise semantic and powerful inference tools. We have proposed a terminology for the PF framework. From this terminology we build a TBOX for PF and a ABOX for a specific problem description. These two elements are then used by DL inference tools to fix a problem into PF. They also give a basis for problem decomposition.

Currently we are working on the implementation of a tool which translates textual or tabular representations of PF and problems diagrams into our DL representation and which performs the reasoning tasks.

Further works address investigations on more elaborated tools for problem decomposition. Another research direction is to investigate software engineering methods with emphasis on domain engineering [14] which is a basis for the specification of the world domain part of a PF.

References

[1] M. JACKSON *Software Requirements & Specifications: a Lexicon of Practice, Principles, and Prejudices*. Addison-Wesley, 1995,

[2] M. JACKSON *Problem Frames*. Addison-Wesley 2001.

[3] D. JACKSON and M. JACKSON *Problem decomposition for reuse* Technical Report Carnegie Melon University CMU-CS-95-108

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

[5] D. BJORNER, S. KOUSSOUBE, R. NOUSSI, and G. SATCHOK *Michael Jackson's problem frames: Towards methodological principles of selecting and applying formal software development techniques and tools.* In 1st IEEE International Conference on Formal Engineering Methods. IEEE Computer Society Press, 1997

[6] S. JON, G. HALL, L. RAPANOTTI, M. JACKSON *Problem frame semantics for software development. In Software and Systems Modeling*, Volume 4 Number 2, pages 189-198, May 2005.

[7] H. PANETTO, N. BOUDJILIDA *Formalizing Problem Frames with ontology. Interoperability for Enterprise Software and Applications:* Proceedings of the Workshops and the Doctorial Symposium of the Second IFAC/IFIP I-ESA International Conference: EI2N, WSI, IS-TSPQ 2006

[8] Xio Hong CHEN, Zhi JIN and Lijun Yi *An ontology of Problem frames for guiding Problem Frames specification Knowledge Science,* Engineering and Management Lecture Notes in Computer Science Volume 4798, 2007, pp 384-395

[9] M. JACKSON *Problem Frames and Software Engineering*, The Open University, December. 2004

[10] F. BAADER, D. CALVANESE, D. MCGUINNESS, D. NARDI, P. SCHNEIDER, *The description logic handbook: Theory, implementation and applications*, Cambridge University Press (ISBN-13: 9780521781763 j ISBN-10: 0521781760), 2003

[11] I. HORROCKS, U. SATTELER S. TOBIES, *Practical reasoning for expressive description logics*, Logic for Programming and Automated Reasoning, pp. 161-180, 1999

[12] S. BRANDT; ANNI-YASMIN TURHAN, *Using Non-standard Inferences in Description Logic. What does it buy me*, Proc. of KI-2001 Workshop on Applications of Description Logics (KIDLWS.01). Volume 44 of CEUR (http://ceur-ws.org/)/. 2001

[13] D. BERARDI, D. CALVANESE, G. DE GIACOMO, Reasoning on UML *Class Diagrams using Description Logic Based Systems* Proc. of the KI.2001 Workshop on Applications of Description Logics. Volume 44 of CEUR Electronic Workshop Proceedings, http://ceur-ws.org/ Vienna, September 18, 2001

[14] D. BJORNER. *Domain engineering* – technology management, research and engineering. JAIST February 2009

Authors' Profiles

KOUSSOUBE Souleymane (1962－), male, Institut Africain d'Informatique Professor, Ph.D., his research directions include ontology, formal methods, knowledge-based systems, Business intelligence.

NOUSSI Roger (1960－), male, Institut Africain d'Informatique Professor, Ph.D., his research directions include ontology, knowledge representation, formal methods, Domain theory.

KONFE Balira Ousmane (1974－), Institut Africain d'Informatique, Professor, Ph.D., supervisor for Ph.D. candidate, his research directions include Bio-mathematics modeling, global optimization, parameters identification technique, intelligence computation and optimal control

How to cite this paper: Souleymane KOUSSOUBE, Roger NOUSSI, Balira O. KONFE,"A Formal Description of Problem Frames", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.4, pp.56-65, 2014. DOI: 10.5815/ijitcs.2014.04.07

