From Process Model to Problem Frame – A Position Paper

Karl Cox
Computer Science & Engineering,
University of NSW, Australia
karlc@cse.unsw.edu.au

Keith Phalp
ESERG,
Bournemouth University, UK
kphalp@bmth.ac.uk

Abstract

Jackson's Problem Frame (PF) approach presumes that some knowledge of the application domain and context has been gathered so that a Problem Frame can be determined. However, the identification of aspects of the problem, and hence, its appropriate 'framing' is recognized as a difficult task. One way to help describe the problem context is through process modelling. Once contextual information has been elicited, and explicitly described, an understanding of what problems need to be solved will emerge. However, this use of process models to inform requirements is often rather ad-hoc. Hence, this position paper proposes guidance for directly deriving Problem Frames from business process models. The paper presents an outline method for PF derivation, and argues why this may be useful to the developer. Finally, the authors discuss the issues involved in attempting to derive a more formal mapping between Problem Frames and business process models.

1. Introduction

In recent years many software developers have produced models of client business processes [1] as an up-stream software development phase [2]. However, although it is generally agreed that such process models are valuable in informing requirements, the exact nature of how the process model maps to subsequent (requirements) phases is less clear.

Some authors have suggested what might be termed 'process approaches' [3] to development methods, but these tend to adopt particular design tactics, where the process model replaces more 'popular' design notations. Others have attempted to examine how process models might map to existing approaches, for example, mapping process models to formal approaches [4] or more latterly, to use cases [5]. Although there is merit in these approaches, one of the problems is that in methodological terms they are implementation dependent. That is, they assume a particular design approach, whether process driven or more conventional (such as the UML) [6].

However, it would be particularly useful if process models could be used to help partition and inform requirements, without assuming a particular subsequent approach to design. This leads on to the idea of combination with Problem Frames [7]. Indeed, one of the premises of the PF approach, is that the proper 'framing' of the problem should suggest appropriate notations both for requirements capture and design [8]. In addition, it is also clear, that whilst simple single frame problems may often be correctly identified, the framing of real-world problems is often far from trivial [9].

Therefore, in this paper, we attempt to show how process models might be used to inform the derivation of Problem Frames. This would then allow process knowledge to be used within requirements phases, and would aid the, non-trivial, process of 'framing' problems. As an exemplar notation, we use Role Activity Diagrams [10] a well-regarded process modelling notation.

1.1. Related work on problem frames

Related work on PFs has focussed on identifying what techniques are most useful to eliciting and documenting requirements and specifications once the PF is known [8, 11], and in attempting a formalization of the PFs [12]. Current research is exploring the role PFs have with aspects of software architecture [13]. These works view the PF as already determined and present ways to help subsequent development. Sikkel et al. [14] propose a variant on the PF. They present a decision tree to help determine what kind of business solution a company might need, such as whether to opt for a COTS product or to bolt on new functionality to the current system. With regard to process modelling and problem frames, there is, to our knowledge, no research currently being conducted.

2. From process models to context diagrams: a good starting place?

The step from process models to context diagrams is not new [15]. Indeed, to take Jackson's variant of the traditional context diagram and map from a Role Activity Diagram (RAD) is straightforward. Table 1 shows the components of both diagrams and how they map.

Table 1. Mapping RAD to context diagram

RAD	Jackson Context Diagram
Role	Domain of Interest / Machine
Interaction	Interface
Action	-

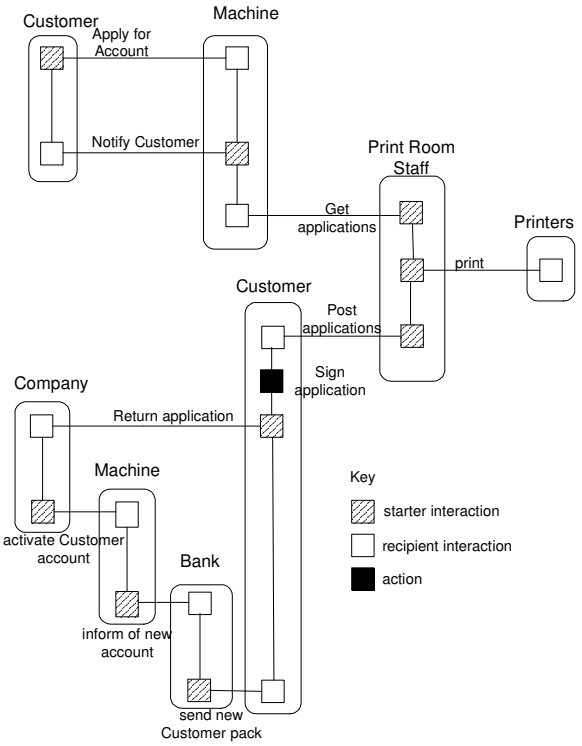

Figure 1. Example role activity diagram

As an example, figure 1 describes a RAD of a simplified process of applying for an online share trading account. This is mapped to a context diagram (figure 2).

Essentially the diagrams (figs. 1 and 2) are the same. In fact, it can be conjectured that there is a loss of information if we describe by context diagram alone. There is no explicit representation of the internal actions of the domains that are vital to the success of the business. In figure 1, actions within roles are made apparent (by black squares) – the Customer role action 'sign application'. What is also required is a textual description of each domain (not detailed in this paper).

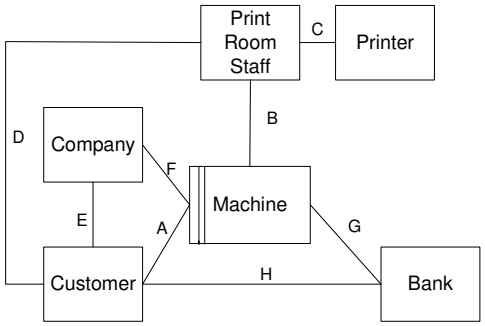

Figure 2. Context diagram

The interfaces between the domains can be made explicit and are described in table 2. For example, for interface A, CU!{…} means that the Customer domain is responsible for the interaction with the Machine domain.

Table 2. Interfaces on the context diagram

Interface	Description
A	CU! {apply} MA!{notification}
B	PRS!{retrieve application}
C	PRS!{print application}
D	PRS!{post application}
E	CU!{return application}
F	CO!{activate account}
G	MA!{new account details}
H	BA!{welcome}

This indicates which domain is responsible for what, that is, what role they play in the process. The next step ought to be to consider how to determine the PFs. But there is a problem here.

3. Problems mapping to problem frames?

The context diagram, as derived from the process model, does not explicitly show the information the problem frames might need. For example, if we have a Workpiece frame, where in the context diagram or the process model is there a design domain (other than the machine)? The process model does not necessarily describe what type of problems there might be – just the way that the business works for this particular scenario.

As such, it is not clear whether we are describing fundamental problem frames or decomposed 'process-

oriented' problem frames. Hence, there is a risk of simply following the process through onto subsequent frames without consideration of the 'big picture'. That is, bypassing the fundamental problem frames for the finer details of transforming a process model into a set of 'process frames'.

4. The frames

What, then, can be derived from a process model that will determine the problem frames? Figure 1 shows the Customer creating an online trading account. Thus, this is a Workpiece problem frame (figure 3).

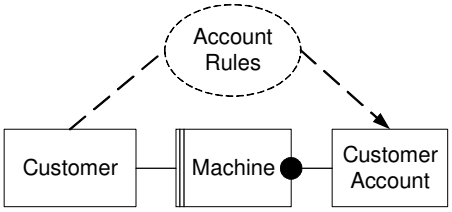

Figure 3. Workpiece frame

It can be seen that the Customer Account domain in figure 3 is not apparent in either the context diagram or the RAD, though it is a fundamental (design) domain in this problem. This shows that the mapping to the context diagram from the RAD, though easy, does not necessarily provide all of the information for the problem frame. (The black dot indicates the Customer Account domain is found within the machine itself – it is a design domain.) However, we can elicit this domain by further exploring the nature of the account creation activity. This can be achieved, for instance, by decomposing the RAD further, revealing the details of the interaction. We can also examine the interfaces within the context diagram. Validating the account creation process with the necessary stakeholders will verify that the Customer Account domain is right – and of its legal status.

4.1. Further potential problem frames

There are at least two other frames identified through further analysis of the problem domain (not shown): the Commanded Behaviour frame allows the Customer to manipulate their Customer Account online – transfer funds, buy and sell stocks and shares. The third frame would be an Information Frame. The Customer can check the current stock prices on the Web Application.

These three core frames might need to be decomposed further. For instance, how does the Web Application show the stock prices? Perhaps a Connection frame is required here.

5. Outline of a mapping

Process models do not necessarily convey the information required to determine PFs, even when mapped into context diagrams, because domains key to the success of the PF approach are not always apparent, particularly if the missing domains are design domains – such as in a Workpiece. This makes the step from a process to a PF view more complicated. We thus propose initial guidelines to assist in this task. The guidelines are rudimentary and are based upon our experiences thus far. We will formalise them as our research continues. Table 3 describes the steps in this (iterative) process.

Table 3. RAD to problem frame

Step	Action
1	Describe RAD
2	Identify outcomes of interactions
3	Identify potential domains from outcomes
4	Identify potential rules that govern interactions
5	Identify problem frames

The first step is to describe a process model (in our case a RAD). We note that companies might have existing process models in other notations, but choose, for now, to limit our guidance to RADs.

Step two identifies the outcomes of interactions between roles. In the above example, an outcome of the 'apply for account' interaction is the creation of a new customer account.

As step three indicates, this outcome is then considered as a potentially new domain. Each is asked:

- Is the outcome something that will be used, altered or referred to a number of times from different perspectives? In other words, a domain of interest. That is, it is not simply a transient outcome. (The Customer Account will be manipulated or referred to through its lifetime by the Customer, the Bank, and the Print Room Staff in different scenarios.)
- We use Bray's domain taxonomy to determine its type [11]. Is the domain a design domain? Inert? (We can say that the Customer Account is something that will be created and held within the machine and will not change its state independently.) Other questions are: is the domain static (not changeable with time in any way), reactive (predictable), completely controllable (programmable), partially predictable (biddable) or entirely uncontrollable (autonomous)?

Step four explores what rules are in place to control interactions. For instance, when the Customer applies for the account, they have to enter required financial information, such as current bank account details. The financial credit status of the Customer, we discover, is electonically checked by connecting to a credit agency. Legal requirements also govern the application procedure and these have to be discovered. The machine then steps the Customer through a precisely defined application procedure.

Step five then identifies the PF. For example, once the RAD is described, the Customer Account has been identified as an outcome of the interactions, and the legal and financial requirements (the rules) are determined. We can state: We have an inert, design domain (Customer Account) – created by the Customer and considered a legal document (legal / financial rules). We therefore have a Workpiece problem frame.

6. Discussion

This position paper has set out to outline a way to derive (appropriate) Problem Frames from process models. The method is illustrated by describing the derivation of a Workpiece frame from a Role Activity Diagram. It is shown that although traditional mapping from business processes to context diagrams might be viable, as an intermediary step towards a problem frame, such a mapping has potential pitfalls because important design domains are often missed. Therefore, we can bypass this step and consider the problem frames direct from the process model. Key to eliciting further domains, vital to the identification of the problem frames, is exploring the interactions between roles for outcomes (potential domains) and rules (potential requirements or contraints governing use or control of the domains).

6.1. Further work and potential issues

Our goal is to provide a complete, formalised set of guidelines to help determine problem frames from process models. However, there are some 'mapping' issues to be addressed.

Is it necessary for a complete mapping to be produced? We are not saying that RADs and PFs are isomorphic, in the way UML sequence and collaboration diagrams are. For example, it is likely that in moving from the RAD to the PF, some information is lost. When changes are made in actual requirements some of these may impact the business model, but (if they do not concern the interfaces between the domains or in the rules governing the frame) the frames may be unaltered. Hence, it may be necessary to consider a multiple mapping among business model, problem frame and requirements. Indeed, similar issues have been described among process models, use cases and class diagrams [5].

This also brings into question whether a direct mapping is most beneficial or whether it may be necessary to use an intermediate notation. Again lessons may be drawn from process modelling where notations, such as POSD, have been used in this manner [2].

Finally, we note, that although well regarded, the existing PFs are seen as a starting point, and that certain contexts may yet suggest the need for further frames.

7. References

[1] P. Henderson, "Software Processes are Business Processes Too", *Third International Conference on the Software Process*, IEEE Comp. Soc. Press, Reston, Virginia, USA, Oct 1994.

[2] K.T. Phalp, "The CAP Framework for Business Process Modelling", *Information and Software Technology*, 40 (13) 1998, pp. 731-744.

[3] Warboys, B, Kawalek, P, Robertson, I. and M Greenwood, *Business Information Systems*, McGraw Hill, 1999.

[4] G. Abeysinghe and K.T. Phalp, "Combining Process Modelling Methods", *Information and Software Technology*, vol. 39, num. 2, 1997, pp. 107-124.

[5] K.T. Phalp and K. Cox, "Guiding Use Case Driven Requirements and Analysis", *7th Int. Conf. on Object-Oriented Information Systems*, Springer, LNCS, Calgary, August 27th-29th 2001, pp.329-332.

[6] Jacobson, I., Booch, G., and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[7] Jackson, M., *Problem Frames*, Addison-Welsey, 2001.

[8] Kovitz, B., *Practical Software Requirements*, Manning, 1999.

[9] K. Phalp, and K. Cox, "Picking the Right Problem Frame - An Empirical Study", *Empirical Software Engineering Journal*, 2000, 5(3), pp. 215-228.

[10] Ould, M., *Business Processes*, Wiley, Chichester, 1995.

[11] Bray, I., *An Introduction to Requirements Engineering*, Addison-Wesley, 2002.

[12] 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", *1st IEEE Int Conf on Formal Engineering Methods*, IEEE Comp Soc Press, Hiroshima, Japan, 12-14 November, pp. 263-270.

[13] J. Hall, M. Jackson, R. Laney, B. Nuseibeh, and L. Rapanotti, "Relating Software Requirements and Architectures using Problem Frames", *RE'02*, IEEE Computer Society Press, Essen, Germany, Sept 2002, pp. 137-144.

[14] K. Sikkel, R. Wieringa, and R. Engmann, "A Case Base for Requirements Engineering: Problem Categories and Solution Techniques", *REFSQ'2000*, Stockholm, Sweden, 5-6 June 2000.

[15] Britton, C. and J. Doake, *Software System Development: a gentle introduction*, McGraw-Hill, 1993.