ELSEVIER

Deriving requirements from process models via the problem frames approach☆

Karl Cox[a,b,*], Keith T. Phalp[c], Steven J. Bleistein[a,b], June M. Verner[b]

[a]*School of Computer Science and Engineering, University of New South Wales, Australia*
[b]*National ICT Australia, Sydney, Australia*
[c]*Empirical Software Engineering Research Group, Bournemouth University, UK*

Abstract

Jackson's problem frames is an approach to describing a recurring software problem. It is presumed that some knowledge of the application domain and context has been gathered so that an appropriate problem frame can be determined. However, the identification of aspects of the problem, and its appropriate 'framing' is recognised as a difficult task. One way to describe a software problem context is through process modelling. Once contextual information has been elicited, and explicitly described, an understanding of what problems need to be solved should emerge. However, this use of process models to inform requirements is often rather ad hoc; the traceability from business process to software requirement is not always as straightforward as it ought to be. Hence, this paper proposes an approach for deriving and contextualising software requirements through use of the problem frames approach from business process models. We apply the approach on a live industrial e-business project in which we assess the relevance and usefulness of problem frames as a means of describing the requirements context. We found that the software problem did not always match easily with Jackson's five existing frames. Where no frame was identified, however, we found that Jackson's *problem diagrams* did couch the requirements in their right context, and thus application of the problem frames approach was useful. This implies a need for further work in adapting a problem frames approach to the context of e-business systems.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Problem frames; Process modelling; Requirements engineering; E-business systems

1. Introduction

In recent years many software developers have produced models of client business processes [1] as an up-stream software development phase [2]. 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 activities 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 methodology dependent. That is, they assume a particular design approach, whether process driven or more conventional, such as the UML [6].

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. It is only upon identification of the requirements problem that
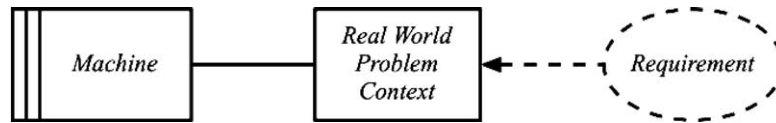
Fig. 1. Elements of problem frames.

anything about the problem is known and as such a predetermined methodology might be inappropriate in addressing the newly understood problem. This leads on to the idea of combination with problem frames [7]. Indeed, one of the premises of the problem frames approach is that the problem 'framing' should suggest appropriate notations for requirements analysis and specification [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], implying that today's increasingly complex systems will be more difficult to frame.

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 the requirements phase, and would aid the, non-trivial, process of 'framing' problems.

One possible objection is that such process models may not already exist. However, our experience with e-business systems (and in particular web-based systems) is that the organisations deploying them are often focussed on process, and often view development of such systems as specifically supporting business processes. Indeed, business models are often viewed as a way of categorising types of e-business applications [10]. For example, in the case study described in this paper, the organisation had dedicated business process analyst roles, and explicitly viewed process models as the starting point for understanding business and, subsequently, systems requirements.

Furthermore, it has been suggested that a key factor in the success of the e-business enterprise is the extent to which system requirements support business strategy [11–15]. Clearly, process models are one way in which this mapping from the strategic business view through to requirements can be supported.

1.1. Introduction to problem frames

Problem frames capture and classify software development problems [7,16]. A problem frame structures the analysis of the problem within its problem space. It describes what is in the real world and how the software is intended to change or guarantee real-world conditions in accordance with the requirements. With its emphasis on problems rather than solutions, the problem frames approach uses an understanding of a problem class to allow the 'problem owner' with his or her specific domain knowledge to drive the requirements engineering process by

selecting the appropriate development method specifically designed to solve the problem type.

Problem frames are a means of understanding and describing the problem context when software will provide (some part of) the solution. As such they are akin to design patterns [17] in that they provide a recognised problem pattern that has a known solution method. However, problem frames differ from design patterns since they represent real world phenomena, as opposed to solution phenomena.

Fig. 1 illustrates some essential elements of the problem frames model. The real world problem context provides us with information about the structure, processes and tasks that are already true of the problem domain. The requirement states which properties we wish to be true given a built software solution, the machine that will work within its real world context. The connection between the real world problem context and the machine is represented by the shared phenomena at the boundary between the problem and the solution. Shared phenomena can be data, events, commands and states. Domains responsible for shared phenomena are described through a syntax [7],

a. DO ! {x}

such that, at interface 'a', domain DO is responsible '!' for {x} phenomena.

For most software problems there will be a number of requirement ovals and a number of domains of interest. Each requirement set connects a number of domains in two ways. An arrowhead indicates that the domain is constrained by the requirement. That is, the machine must guarantee that the state or behaviour of that domain satisfies the requirement. A requirement reference, with no arrowhead from requirement to domain, indicates that the requirement refers to some phenomena in that domain [7]. Domains can appear a number of times in the problem diagrams and problem frames through the principle of *projection*. A requirement might be interested in only certain phenomena or certain behaviour of a domain, given the particular sub-problem addressed. In a different projection, other phenomena in the same domain might be of interest to the requirement for another particular sub-problem.

Jackson describes two moods, in the grammatical sense, to represent the problem context and the requirement [7]. Indicative mood represents everything in the problem context that is given and will remain unaffected

by the machine. Optative mood represents the way we would like everything to be, given the construction of the machine. This is the requirement. A requirement can change the properties, states and behaviours of domains of interest but cannot affect indicative properties. In this way, we can get three descriptions. (1) The problem context, everything in the application domain that is relevant to understanding the problem and the scope of the requirements. (2) The requirement, all that we would like the machine to bring about in the problem context. (3) The specification, which describes the shared phenomena that directly connects the problem context to the machine in order to achieve the requirement.

A problem diagram, containing the same elements as in Fig. 1, describes a software problem showing the problem parts consisting of problem context and the requirement. Problem frames are derived through decomposition of problem diagrams. Even though the software/hardware system may consist of multiple devices or computers, for the purpose of a problem diagram these are represented as a single machine. Decomposing problem diagrams reveals a greater level of detail, including separate distinct machines.

Jackson suggests five basic problem frames [7], though states that this is not an exhaustive list. The problem frames are:

- Workpiece frame, for example, a Petri net tool or text editor.
- Commanded behaviour frame, for example, a lift controller system where a human interacts with the lift controller to call the lift, select which floor to travel to, etc.
- Required behaviour frame, for example, a simple, time-driven, traffic lights system.
- Information display frame, for example, a digital speedometer display on a car dashboard.
- Transformation frame, for example, a file conversion tool such as a PDF writer.

In an earlier work, Jackson [16] suggested a connection frame. This might be a gas sensor monitoring oxygen levels in a mineshaft safety system, the sensor being the connection between the actual oxygen and the information system reporting the levels of oxygen. However, this frame was not included in his 2001 book. Jackson has re-labelled it only a connection domain.

A key concept of problem frames is that the requirements are not always at the interface of the machine. The requirements are in the application domain and without explicitly describing the application domain, the problem context, it will be very difficult to identify the *right* requirements. But a problem's characteristics will not be recognised until the problem context has been explored and described. As such, adherence to only one particular methodology regardless of the problem's characteristics is not recommended.

1.2. Related work on problem frames

Related work on problem frames has focussed on identifying what techniques are most useful to eliciting and documenting requirements and specifications once the problem frame is known [8,18], and in attempting a formalisation of the problem frames [19]. Current research is exploring the role problem frames have with aspects of software architecture [20,21]. These works view the problem frame as already determined and present ways to help subsequent development. Sikkel et al. [22] propose a variant on the problem frame. 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. Nelson et al. [23] propose a suite of possibly finer-grained problem frames that specifically focus on the geographic applications domain. It is unclear whether these are really different problem frames or variants on Jackson's originals. Lin et al. [24] address security threats by proposing what they call abuse frames. These appear to be at a lower abstraction than Jackson's original frames and address specific non-functional requirements that might arise in control or information problems. Bray and Cox [25] have proposed a further fundamental problem frame for simulation problems. There has even been a proposal to use problem frames as the metaphor in extreme programming [26]. These works do not consider process modelling at all. Indeed, there is, to our knowledge, no published research on connecting process modelling and problem frames, save for the original proposal which forms the basis of this paper [27], and combined with goal modelling and Jackson context diagrams to represent the optative and indicative states combined of an e-business system problem description [13,14].

1.3. Domain modelling

Domain modelling in the sense of applying recognised, accepted solutions draws similarities to the world of design patterns [17]. Essentially, domain pattern models are reusable patterns of problems. The problem frames approach presents patterns that are wholly problem focussed yet frame the problem with a known solution. There are other domain modelling approaches that present similar problem patterns and these vary in terms of abstraction and context. For example, Rubenstein and Waters [28] introduced what they called clichés for seven distinct problem domains. At a different level of abstraction, Maiden and Sutcliffe [29] present object system models from the NATURE project. They defined 200 of these, each representing different problem domain elements. Maiden and Hare [30] show ways to determine which problem category one requires—from those defined in the NATURE project—by using a card slot approach. Coad et al. [31] presented a number of Coad-and-Yourdon object patterns

and strategies for analysis modelling; these though appear more at home in early design than domain modelling. Fowler [32] presents a large number of analysis patterns, described in a hybrid object-oriented modelling notation specifically for the financial domain. Robertson [33] describes how requirements process patterns can be derived from use cases/events to enable requirements chunking for systems construction with tight focus on the business problem. We are unaware as to the uptake and/or success of any of these approaches in an industrial setting. In other closely related areas, this kind of problem, as opposed to solution, pattern approach is showing itself to be of importance to wider business success, for instance, at the enterprise architecture level [34], and standard as well as e-business patterns [10,35,36].

1.4. Role activity diagrams

As an exemplar notation to describe process models we use role activity diagrams [37], a well-regarded process modelling notation. A role activity diagram (RAD) is used to describe business processes that can involve actions and interactions among roles. Roles can be humans, departments and organisations, as well as software and hardware systems. A RAD provides an excellent means of describing dependencies between roles in organisations that work discretely and in unison to achieve a goal. A RAD has various components, the most common of which are illustrated in Fig. 2.

All roles start in an initial state. For example, role A starts in an initial state and then has an event, an action, 'do work', which is independent of other roles. On completion of the work, the role would be said to have moved to a new state of work completed. Although states are often omitted, as in Fig. 2, a formal view would be that the event, and action of role A, has a pre-state of 'initial' and post-state of 'work completed'. (We have kept the states intact in our case study process model, found in Fig. A.1.)

Some work is then delegated to a colleague. This is a shared event. Although the mechanism of delegation is immaterial, the result is that both roles involved move to the state of work delegated. These shared events are termed

Fig. 2. Elements of a role activity diagram.

interactions. Although there is no sender and receiver as such, role A is said to initiate (be the active role) whilst role B is passive in this interaction. Role B is then in a state to independently 'do work'. Role B then 'returns work to colleague', role A, who is in a state to receive it and so on.

1.5. Purpose of this study

Problem frames are becoming an established approach within the academic requirements engineering community, witness the first International Workshop (IWAAPF) at ICSE 2004 [38]. The approach is now taught as part of undergraduate and postgraduate requirements engineering at several Universities, and forms part of an analysis and design method, called problem domain oriented analysis [18]. Despite this, to our knowledge, there are no published industrial studies on the use of problem frames. There are no studies that attempt to apply problem frames to a complex domain, such as an e-business system. Therefore, we had no idea whether it would be useful to apply problem frames to a real industrial project, nor how easy it would be to apply. This led to our first research question,

RQ 1: Can the problem frames approach be applied to complex, industrial projects?[1]

Those studies of problem frames that have been published, as reported in Sections 1.1 and 1.2 and also in [38] with the exception of [12,14,15] either describe simple theoretical case studies, or small, well understood, classical software engineering exemplars.

Problem frames examples, as discussed by Jackson in his books [7,16], and others [8,18] have simple starting places to begin a problem domain-oriented analysis. However, complex systems in real industrial settings rarely have things so easy. This makes it difficult to know where to begin a problem frames analysis and leads to research question 2,

RQ 2: Where is a good starting point to begin a problem frames analysis in the context of the case study domain?

A simple and entirely practical approach was taken to research question 2. We took the developed process models as our starting point, or in other words, our problem boundary, because this is where our customer saw their problem boundary. They presented us with very simple process models that described the basic process that we have extended and refined, as described in this paper.

We worked on the project as contracted requirements engineers as part of the development team, not as academics conducting an isolated study. Thus, the subject matter
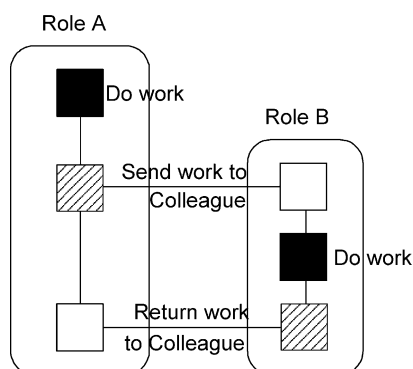
---

[1] Note that better results might have been obtained through a different approach. However, our study addresses problem frames and not another approach, so we can only comment upon the success of this particular case.

reported in this paper is from a live, real world project that addressed a real business problem. Thus, in this paper we set out to apply problem frames to a complex industrial context, that of development of part of an e-business system. For pragmatic reasons we considered that one way in which this may be aided is to use existing process information, which we present through role activity diagrams, to guide the framing of the problem by providing a starting point for describing the problem context.

The major contribution of this paper is that it is, to our knowledge, the first reported application of problem frames in an industrial project. The paper also contributes a potential starting place to begin a problem frames analysis—process modelling—and means by which to connect process models to problem frames. The paper critically evaluates the problem frames approach, indicating that other problem frames might be required for e-business systems on top of the existing frames. Another contribution is the derivation of requirements from process models.

The paper takes the following form. Section 2 introduces the industrial case study, thus placing this work in the context of a re-engineering problem for what is an e-business system. Section 3 briefly discusses equivalence of role activity diagrams to context diagrams. Section 4 then presents our proposed approach for mapping to problem frames. Section 5 provides descriptions of that mapping in the case study. Section 6 discusses validity threats to the approach. Section 7 discusses the approach and the implications for problem frames in light of the findings. Section 8 draws conclusions.

2. Background to the industrial study

The case study involves the *re-engineering* of one important process for part of a wider e-business system built initially by a small start-up company but now owned by a global financial organisation. We define e-business as the 'marketing, buying, selling, delivering, servicing, and paying for products, services, and information, primarily across nonproprietary networks, in order to link an enterprise with its current and target customers, agents, suppliers, and business partners' [10]. An *e-business system* enables this. In our specific case, the e-business system as a whole is a financial system that allows customers to buy and trade stocks and shares, as well as conduct other financial activities, across the Internet. The financial partners are the company itself (now a global organisation), a UK high street bank, a stock brokerage company and a credit checking agency. Our case study is a real world example in that we conducted contractual work on the project to address the company's requirements problem. We were thus working as engineers rather than academic researchers.

To understand where the major problem areas for the newly proposed system lay, we first developed role activity diagrams (see Fig. A.1). The process models were derived from simpler process models developed within the company by a business process analyst who was also acting as project manager on this project. The role activity diagrams were also derived from requirements elicitation activities, such as interviews, observations, company documentation, formal validations and informal communications with stakeholders. The process models were correct and consistent with the needs of the company and were as complete as considered necessary by those working on the project, such as the project manager, development manager, systems analyst and company vice-president. Full validation of the process models can be found in [39]. We had to understand the system's use from two viewpoints, (a) potential customers applying for accounts and (b) the print room staff as they access the system in order to print and post application packs to customers. To further define what kind of sub-problems we might have, we decided to use problem frames. As has been outlined in the Introduction, we saw that problem frames had significant potential, but we were concerned that we might have problems applying the approach on a live project effectively. Therefore, we chose to use the process models to guide the use of problem frames (research question two). This makes it necessary for us to explicitly state how the process model would be used, and then to attempt such an application (Sections 4 and 5). Our experience of forming, but mostly using, such an approach within the industrial context is reported in the remainder of this paper. However, we now give further background to the specific problem that we wished to tackle.

The software company itself is somewhat typical of its type in that documentation is kept to its barest minimum [40]. In our case, the company had no requirements or design documentation. However, we did have full access to the system and its test site, which allowed us a deeper understanding of the product itself. As has been stated, we did have access to data describing the process.

The existing system is an online, real time, financial product that allows customers to open stock market trading accounts, later housed at a high street bank, buy and sell stocks and shares in real time and also to open/transfer government bond investment accounts. The original system contained three servers, and an unstable, poorly designed access database meaning a number of unnecessary interfaces between the various elements of the system—legacy servers that added unnecessary complexity to the process. Our first task was to try to understand the purpose of the product so that a re-engineering of its processes could be achieved.

A further problem was that legal requirements, at the time of the project, forced the company to obtain *handwritten* signatures from their prospective customers. As such, the company has an in-house print room that prints individual applications. These are called pack types, where different types of application are labelled under different packs; the print room can only print one pack type at a time—an aim is to reduce the different types from 21 to 2,

making the printing process easier. Pack types contain all relevant information about the customer for the account type created, for sending to the customer for immediate signature. One important aim was to improve the lot of the print room employees by making the task of printing applications as simple as possible. Our specific remit was to redesign the system around the following three constraints:

C1. The print process must be able to print and post up to 1000 customer applications per day.

There is no proposed increase in printers or employees. As such, the software design will have to ease the print process by reducing pack types and make printing of different packs simpler.

C2. The selection of printing jobs must be more efficient than the current system.

The current process is very labour intensive and open to much human error. It has been suggested that the generation of print jobs and the selection of print trays should be automated. Automation is perhaps a long-term goal and not part of the suggested solution in this project.

C3. The software design of the application process has to facilitate more effective generation of pack types to allow easier printing of applications.

This constraint relates to the internal design of the system and is beyond the scope of this paper. As such it will not be discussed further here, though [39] provides a complete description of the case study.

3. Process models and context diagrams: equivalence of information

Many companies have process models to describe their systems. However, within e-business the process view is even more vital, since for many the e-business system directly supports the business process. Indeed, for some organisations the e-business system may even be their core business, and for others the development of such systems forms an integral part of their business strategy. For example, within the organisation described in this case study, business analysts had a dedicated process modelling role. Therefore it seems appropriate to understand the problem surrounding the re-engineering task by describing

Table 1
Mapping role activity diagram to context diagram

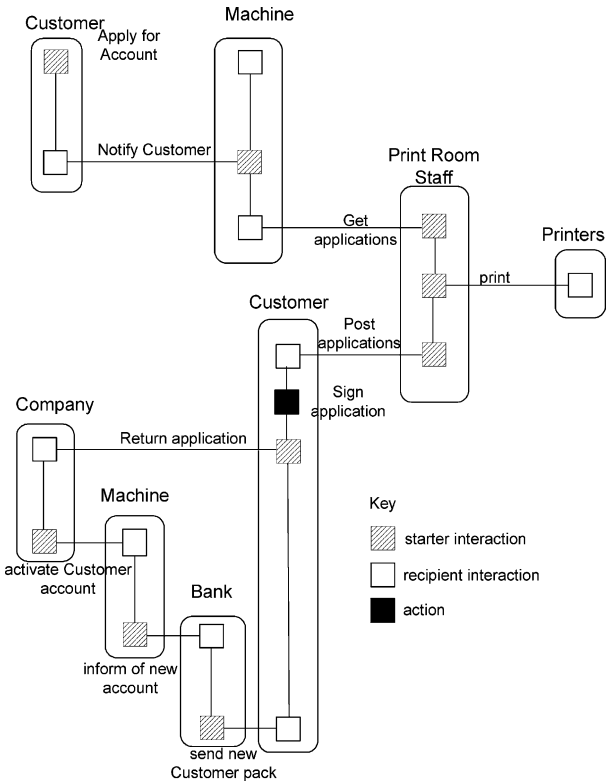| Role activity diagram	Jackson context diagram
Role	Domain of interest/machine
Interaction	Interface
Action	–

Fig. 3. Simplified role activity diagram of actual product process model (see Fig. A.1).

context diagrams from the process models because according to the problem frames approach [7] describing context diagrams is the first step to identifying problem frames. Indeed, moving from process models to context diagrams is a recommended approach [41]. Such a mapping is straightforward. Table 1 shows the components of both diagrams and how they map. However, the information described in a Jackson context diagram [7] is similar to that described in a process model, just described in a different way.

As an example, Fig. 3 describes a simplified role activity diagram for the process of applying for an online share trading account; the unabridged role activity diagram is in Fig. A.1. This is mapped to a context diagram in Fig. 4. Essentially, Figs. 3 and 4 are the same. It can be seen that
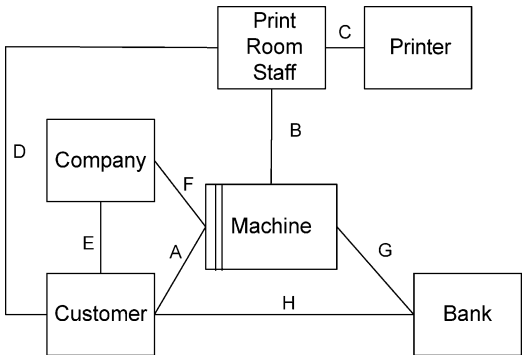

Fig. 4. Jackson context diagram.

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}

Fig. 5. Process model of proposed approach.

there is no explicit representation of the internal actions of the domains that are vital to the success of the business in the context diagram, though these might be derived from an exploration of the interfaces between the domains of interest as shown in Table 2. Since we already have a process model, we consider this sufficiently rich to be taken as good description of the problem context—we have no need to repeat ourselves through context diagrams. So rather than starting with context diagrams we can utilise existing process models, or produce such models, to feed into the framing.

4. Mapping from role activity diagrams to problem frames

We propose initial guidelines to assist in the mapping from role activity diagrams to requirements, which we couch in the particular context of a known recurring problem, called a problem frame. These guidelines emerged as the project was conducted. They therefore evolved over the course of the project and the refined version is presented here. Though we would like to assert the approach's generality, we cannot since it was devised during the actual case study and has only been tried on the case reported in this paper. We can therefore only couch examples in terms of the context of the problem we were attempting to solve. The activities within this iterative approach are briefly described:

0. Explore the problem context.
1. Produce (or revisit) process model (as role activity diagrams).
2. Identify outcomes of interactions.
3. Identify domains from outcomes.
4. Identify potential rules that govern interactions.
5. Identify problem frames.

These are not strictly sequential activities, though some constraints apply. Hence, we take it that step 0 has either already happened or must happen first and that step 1 is dependent upon it. Similarly step 2 is dependent upon step 1, and step 3 upon step 2. Step 4 clearly involves much interaction with step 2, though technically could proceed as
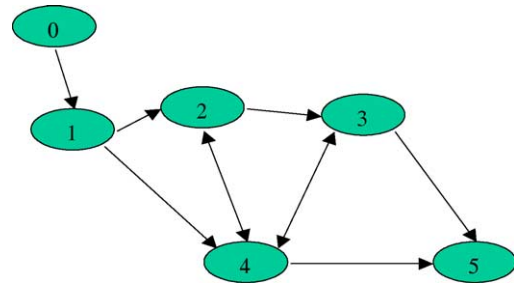
a parallel thread. Step 5, the final step of the suggested approach, is, of course, dependent upon all prior steps. Fig. 5 represents the suggested approach as a simple process flow diagram.

The following sub-sections describe each activity (or step) with an example from the case study. Section 5 then describes further application of this approach to our industrial case study.

4.1. Step 1. Describe role activity diagram

Of course, we have just argued that the great advantage of utilising process models is that the context has already been described (step 0). Thus, it is taken that such a description may already exist. However, it may be necessary to revisit existing process descriptions, and rationalise these into another model, as we did on the project. We find that role activity diagrams not only provide useful process guidance, but may also be used to guide identification of frames. We note that companies might have existing process models in other notations; the notation is not important as long as the information captured in a process model is equivalent to a role activity diagram. To begin this transformation from a complex process model is difficult. Therefore, it makes sense to partition the process model and at the same time abstract away the finer details in order to identify the core problem elements. However, it is important to have established a detailed process model to act as a check that all 'vital elements' of the problem have been considered. Of course, abstracting away the hard or confusing parts of the problem will only lead to an inadequate delivered system.

4.2. Step 2. Identify outcomes of interactions

Step two identifies the outcomes of interactions between roles. If we start at the beginning of the process (though this is not necessary), we have identified the interaction 'apply for account' since this is decomposed into a number of specific events that the customer has to complete in order to open an account. Exploration of this interaction suggests
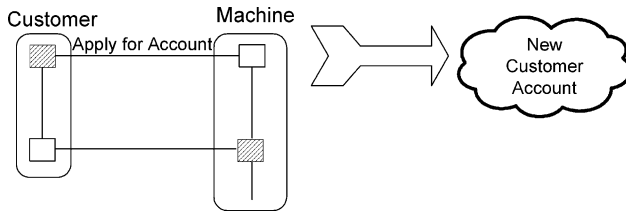
Fig. 6. Possible outcome of interaction.

that the outcome is likely to be a new account application, or in this case study, new account creation, as shown in Fig. 6.

4.3. Step 3. Identify potential domains from outcomes

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 persistent, domain of interest. That is, it is not simply a transient outcome. In this application, a 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 (Fig. 7) to determine the type of domain we are dealing with [18]. This will guide us in what problem frame this might fit.[2] In the above example, we can ask: does the identified customer account domain change with time? The answer is yes, since its state is updated with each transaction that occurs. Therefore, it is not static but dynamic. The next question is, does the domain change itself? For the customer account, the answer is no. It can only change at the behest of the customer, for example, through transactions, and the bank by initiating direct debits, setting interest rates, etc.

The types of domain shown in Fig. 7 are briefly described with examples:

- Static. This might be a CD-ROM of an encyclopaedia. Its state does not change over time.
- Inert. This state refers primarily to software files. Thus when we have an inert domain typically we are referring to something housed inside the machine: a design domain (in the above example, a customer account).
- Reactive (predictable). An abstract data type can be classed as reactive because it needs external stimuli to alter its state but is self-modifying.
- Programmable. Most software applications are programmable.

---

[2] David Garlan presents a slightly different domain list to the one we use (he also considers multi-dimensions and whether domains are tangible or intangible) that could also be appropriate [42].
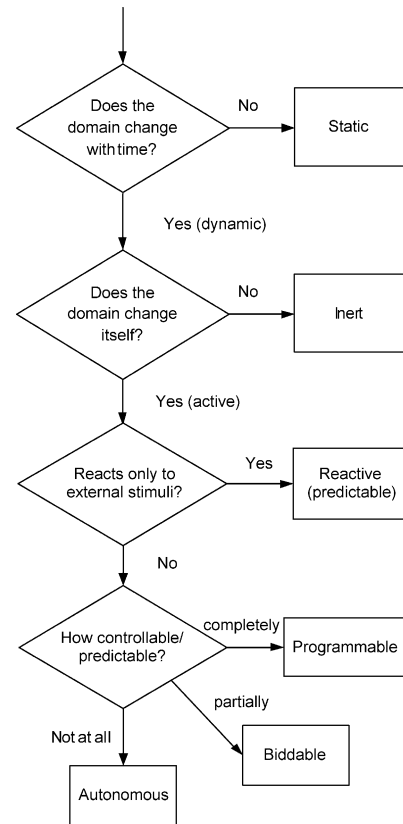

Fig. 7. Domain taxonomy (adapted from Bray p. 99 [18]).

- Biddable. Human beings can be described as biddable since they typically perform the required tasks but do not necessarily do this. It is up to the human to follow a command or not.
- Autonomous. These are domains that are self-governing, such as the weather. They cannot be controlled and are not entirely predictable, though they can be simulated [25].

Knowing what type of domain we are dealing with will help constrain the choice of frame to fit the problem into.

4.4. Step 4. Identify potential rules that govern the interactions

Step four explores what rules are in place to control interactions—these are the requirements. For instance, when the customer applies for the account, they have to enter required financial information, such as current bank account details. Essentially, requirements are derived from interactions in the process model and by an exploration of the outcome of those interactions. However, this is not always obvious. Since the financial domain is governed by specific laws, companies working in this domain will have what is often called a compliance department that makes sure all software and, indeed, all requirements for the proposed software are legal. Exploration of the process

Table 3
Charting the outcomes of the derivation process

Interaction	Roles	Outcome of interaction	Domain properties	Requirements	Problem frames
Register	Customer, web machine	Registrant account	Static	R3. Customer enters particulars	–
Apply for account	Customer, web machine	Customer account	Inert	R1. Customer enters particulars. R2. Customer enters NINO (dependent on account type)	Workpiece
Acceptance notification	Web machine, customer	Text message	Static and transient	R4. The customer should be notified of the success of their application attempt immediately	–
Check credit	Web machine, credit checker	Data	Static and transient	R5. Credit worthiness must be automatically checked through an official credit agency; the response should take less than two seconds R5.	–
Send credit status	Credit checker, web machine	Data	Static and transient	R6. If a negative response, then display Sorry message.	(if R6) Information display frame
Access pack codes	Print room staff, web machine	Access commands (events)	Transient	R7. Print room staff access the web machine to locate the file ready for printing.	–
Prepare for word	Print room staff, web machine	CSV file	Inert	R7.	–
Transfer to Word	Web machine, Word	CSV file	Inert	R8. The CSV file has to be transformed into a printable, standard representation	Transformation (Word mail merge solution)
Organise materials	Print room staff, printer	–	–	–	–
Print	Print room staff, Word, printer	File for printing	Static	R9. Pack types are printed so application forms can be sent to customers for signing	Transformation (printer driver solution)
Collect printed materials	Print room staff, printer	Applications	Static	–	–
Send for posting	Print room staff, post office	Applications	Static	R10. Applications are immediately sent to customers for signing. (time constraint on signed, returned applications of 10 working days)	–
Send forms to customer	Post office, customer	Applications	Static	R10.	–
Return forms	Customer, post office, company admin	Applications	Static	–	–
Update machine	Company admin, web machine	Customer account	Inert	R11. The customer's account can only be activated upon receipt of a completed and signed application form, within the specified time frame	Workpiece (is one requirement sufficient for a problem frame?)
Email/SMS new trader	Web machine, customer	Email/SMS text message	Static	R12. Automatically notify the customer upon the successful activation of their account	Connection frame (if an SMS)
Inform bank	Web machine, bank	Customer account (as an email, probably)	Static (as is a copy of account)	R13. The bank is automatically given all newly activated account details	–
Acceptance message	Bank, web machine, customer	Data, email	Static	R14. The company must inform the customer of the bank's decision to either accept or reject the account	Connection frame
Send account material	Bank, customer	Account materials: cheque book, banker's card, welcome pack, etc.	Real world (do not fit the taxonomy since they are not software related)	–	–

model would prompt an analyst to consider if that under study had to comply with legal regulations and laws. Other requirements, such as some of those represented in Table 3, come from standard requirements elicitation techniques conducted while working on the project: interviews and workplace observations with stakeholders as part of the process model verification and validation process. Example requirements are now described,

R1. A new account is set up by the customer upon provision of all relevant banking information. We need to

know:

    Customer full name;

    Customer current address;

    Account type required (see RXX for account types and their necessary attributes);

    Current bank account details…

R2. The customer must provide their national insurance number (NINO) for opening a (UK) government bond account.

    R2.1 The NINO must be verified for its authenticity.

The machine steps the customer through a precisely defined application procedure. Failure to provide the required information will cause a halt in the application procedure. Providing false information is against the law, and it is a legal requirement that the company notify the customer of this fact, which might be requirement R2.2, for instance.

4.5. Step 5. Identify the problem frame

Step five then identifies the problem frame. From the above example, the new customer account has been identified as an outcome of the interaction 'apply for account'. The outcome is then assessed as to whether it is persistent or transitive. If it is persistent then it will be classified under the domain taxonomy so that we can understand what kind of domain it is and where it fits with potential problem frames. A further test of the permanence of the domain is to see whether it meets the requirements—the rules that govern this domain. In our example, this would be, among others, legal requirements, for example, providing a national insurance number for certain types of account, financial requirements, providing all the necessary information the account procedure demands, for example, the customer's current bank details and interface requirements, guiding the process of the online application. Once the above are determined, we can state: We have an interaction that produces a customer account domain, which is persistent and inert, reacts to external stimuli and is housed in the machine, and satisfies the requirements that govern it. We, therefore, have a workpiece problem frame as shown in Fig. 8.

In complex process models it also makes sense to organise the domains, the interactions, outcomes of interactions, the requirements and any problem frames into a table. In Table 3 the second entry refers to the above example.
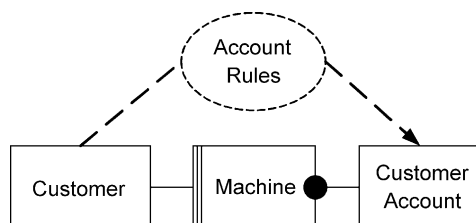

Fig. 8. Workpiece frame.

5. The problem frames: examples of the approach

Further exploration of the interactions in the process model will make this clearer, and will also provide some answers as to the use and usefulness of the approach and of problem frames themselves. Assuming a process model has been developed, the next subsections describe the approach in action for deriving problem frames. We limit ourselves to a number of examples here for reasons of brevity, though Table 3 provides details of all requirements and subsequent frames derived from the process model.

5.1. Credit check

The interface connects the web machine and the credit checker. The credit checker agent is a machine that rapidly assesses the credit worthiness of the customer from the information passed by the web machine. The outcome of the interaction between the web machine and credit checker is a data packet containing the customer's information. This is a point where it might be easy to misuse the problem frames approach. For instance, if one explores the requirement, R5, that the credit checker is sent the customer's data and that the credit checker then returns a 'credit worthy' or not status, it is obvious that this interaction is about the customer and the credit checker. A problem diagram might look something like Fig. 9.

The first thing to note is the introduction of the customer domain to the problem. The requirement is, after all, about establishing the credit worthiness of the customer. The requirement reference 'a' assumes the required problem domain data needed to fulfil the task is provided by the customer. This is provided to the web machine through the specification interface 'c', by means of the events that govern the entry of the data into the web machine. The requirement 'b' constrains the credit checker to perform the correct credit check and to provide accurate feedback, all within a real time constraint of 2 seconds, R5 in Table 3. The interface between the web machine and the credit checker, 'd', provides the correct format for data sent and the response that it subsequently receives. Since this meets the requirements, the next question is to ask what kind of problem frame this is.
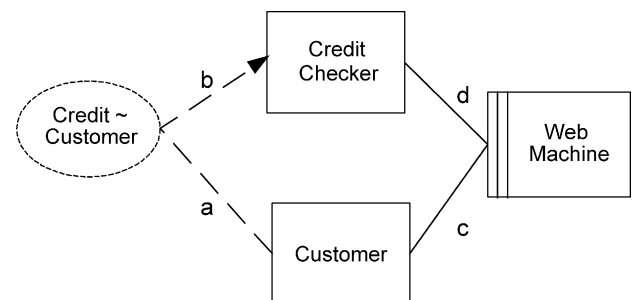

Fig. 9. Possible problem diagram for 'check credit' and 'send credit status' interactions.

- Is it a workpiece? Where is the inert domain, created by the customer and housed within the machine? There is no such domain.
- Is it a transformation problem? It is certainly the case that the input to the credit checker is a lexical representation of some information. However, no transformation takes place. The information flows into the credit checker but is not transformed into anything.
- Is it a control frame? We have a customer, therefore, this must be a commanded behaviour frame? But what exactly is being controlled? The customer is entirely oblivious to the credit checker. Indeed, the credit checker is a software machine that we only interface with. All we do is feed it information and await a yes/no response.
- Is it an information problem? Yes, but is it an information display frame? It is an event–response problem, but with only one event and one response, where the event asks for information based upon the data in the event parameters sent. This is in essence what all information systems are, though the task is the same each time: provide credit information to perform a check. It is only the information that changes with each new customer. We can describe this therefore as an information problem that will become an information display frame if, and only if, there is a negative response. See Section 5.2.
- Might it also be a connection problem? Yes, there is a connection issue but this is a secure socket connection between the web machine and the credit checker. This does not fit as a connection problem frame.

This problem does not fit any of Jackson's problem frames. Therefore, we question whether we might need a new problem frame that is appropriate for this specific problem, all too common in this domain.

5.2. Send credit status

This is the response to the previous interaction (check credit in Section 5.1) and it, too, is an electronic message only. If it is a positive result for the customer then there is no frame here since the customer is not informed of their success, only that the application procedure continues. However, if there is a negative response, R6, then some

information has to be displayed to the customer. This is an information display frame. The problem frame looks like Fig. 10.

The requirement for the credit checker 'e' is to determine the accuracy of the customer's credit worthiness. This is beyond the scope of our problem so we can ignore it here; as such it is referenced, not constrained, that is, no arrowhead. The credit checker passes its *negative* response to the web machine 'f', which, in turn, is displayed for the customer to see 'g'. The customer display is constrained to represent the correct information 'g' from the credit check, hence the arrowhead.

5.3. Prepare for word

This interaction is all about preparing to transfer the selected CSV file to the Word application. The outcome of the interaction is the selection and placing of the particular CSV file, whose selection is dependent upon the pack type selected for printing, in a position or state so that it might be printed by Word, though Word is not part of this particular interaction problem. Is the CSV file a domain of interest? It is certainly manipulated over time and contains the information necessary to enable the correct printing of applications. What kind of domain is it? According to the taxonomy it is an inert domain. Its state does change since its position in the printing process changes when an external entity manipulates it; its representation might also alter at some point in the printing process. Does the outcome meet the requirement? Yes. Since R7 is about locating and selecting the right CSV file for printing, the outcome clearly meets the requirement. Is there a problem frame here? Fig. 11 describes the problem.

The CSV file is a realised domain—it is internal to the machine. This is denoted by the stripe on the left of the domain box; a realised domain can also be represented a black dot from the machine to the domain, as in Fig. 8. The interfaces 'h' are the events and actions that the print room staff commits at the interface of the machine to locate and select the CSV file. The requirement constraint and specification 'i' states that the web machine must make the CSV file visible and selectable to the print room staff. But is this a problem frame? It could be a variant of the commanded behaviour frame where the CSV file is the domain being commanded. As an inert domain it can be
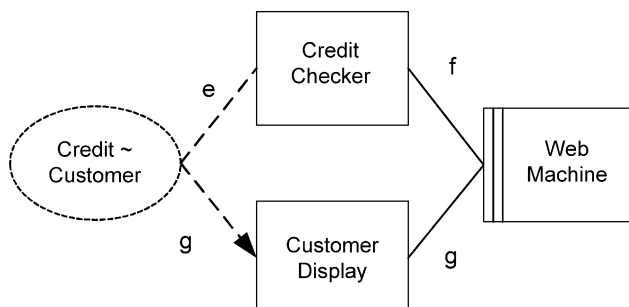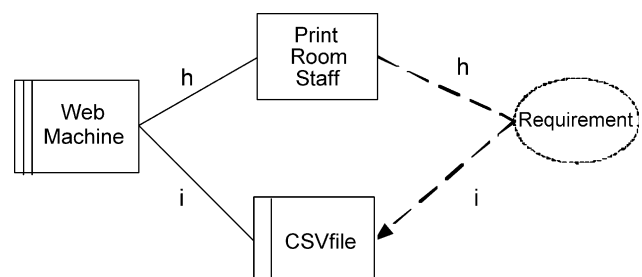

Fig. 10. Information display frame.

Fig. 11. Possible problem diagram for 'prepare for word' interaction.

controlled since its state can change. But since it is internal to the machine and not some external reality, it seems that a problem frame does not fit in this instance.

5.4. Transfer to Word

The transfer of the CSV file to Word is done on the actions of the print room staff so is a continuation of the previous interaction. The outcome of the interaction is listed in Table 3 as the CSV file. This is the object of the interaction, rather than the outcome. However, it is a close enough approximation. We know that the CSV file is an inert domain. What is the requirement? R8: the CSV file has to be transformed into a printable, standard representation. Since this is a re-engineering problem, many of the solutions are already known. In this example, it is known that Word can mail merge certain document types, in our case the CSV files, and therefore the solution presents itself without any effort on our part. There is the possibility of a transformation frame. The CSV file acts as the input domain—it is static and lexical, which fits the properties of the transformation problem. The output is a merged document and it must meet the requirement of representational format, that is, the data in the output domain, a printable Word document, must map exactly to the data of the input domain (CSV file). The transformation machine, in this instance is the Word application. The transformation function, mail merge, is not automatic; user events will govern the mail merge. The transformation frame looks like Fig. 12.

5.5. Print

This interaction involves three roles: (1) print room staff, who give the command to print applications, (2) the Word application, which receives that command and then processes it until the outcome is a file sent to (3) the printer, which receives the print file, queues it and prints it. The outcome of the interaction between the print room staff and Word is simply a shared event: the print command. An outcome of the interaction between Word and the printer is a file to be printed. The printer driver will convert the file into a printable format. The requirement is simply that the pack type selected for printing is printed, as stated in R9. This is a transformation problem though it is unlikely that printer drivers are programmed from scratch any more it is still
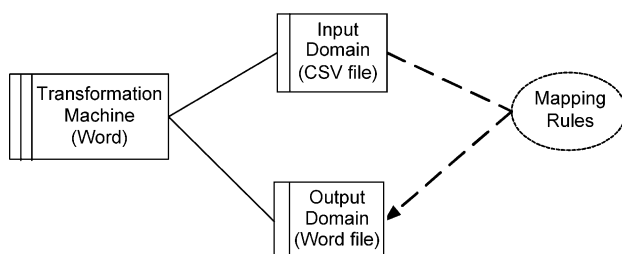
important to identify the problem frame. The outline of a transformation frame is shown in Fig. 12.

5.6. Send for posting/send forms to customer/return forms

These three interactions are combined because they are part of the same sub-process for our purposes. The print room staff pass the printed applications to the post office for distribution to customers. These are important interactions because there is a time constraint on how long the customer account can remain 'live' without being formally agreed to by the customer. Requirement R10 states that applications are immediately sent to customers for signing with a time constraint on signed, returned applications of 10 working days. Thus we should be concerned with the efficiency of the post office as a possible hindrance to the delivery of the applications. The return forms interaction is again dependent upon the post office, not shown in Fig. A.1, so time is an issue. However, since this is part of the problem domain removed from the machine there is no standard problem frame to describe. It is uncertain whether to impose a requirement upon the customer since their action is, at best, biddable. The accompanying documentation clearly states that if the customer wants their account activated they must sign and return the enclosed forms within 10 working days of setting up their account—the date of which is supplied in accompanying documentation.

5.7. Update machine

Upon receipt of a signed application form from the customer, the company's administrators have permission to update the status of the customer's account to one of 'active'. This means that the customer can now start to trade stocks and shares or invest in government-guaranteed bonds. The outcome of the interaction is the activation of the customer's account, which we know to be inert—its state changes but only at the command of the customer, company or bank. Therefore it is a domain of interest. The web machine carries out the updating process on the command of the company administrator. The two roles in the interaction are also domains of interest. The requirement states that the administration staff activates

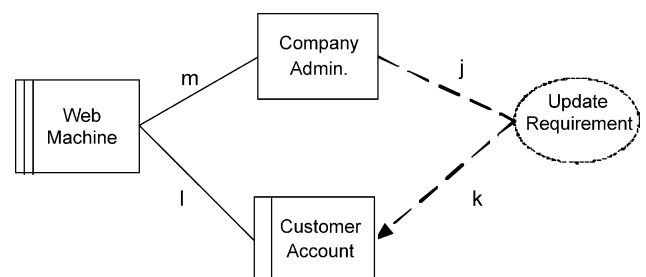Fig. 12. Transformation frame for mail merging the CSV file.

Fig. 13. Problem diagram for updating the status of the customer's account.

the customer's account so that trading can commence, only upon receipt of a correctly completed and signed application form received within the stipulated time frame of 10 working days from the account being opened, R11. What is the problem frame? The problem contains these domains: customer account, company administrator and the web machine. The problem diagram is shown in Fig. 13.

An inspection of the requirement shows that 'j' consists of events that the company administrator does to drive the updating of the account. Interface 'k' is the requirement constraint—that the customer account be activated, and nothing else. Specification interface 'l' is the command from the web machine to the customer account to change its state. Interface 'm' is the commands used by the company admin to make the update and also the feedback given by the machine to the admin.

What kind of problem frame is this? It appears to be a workpiece problem, since the customer account's state is now altered. However, there is only one requirement. The question then arises, can we have a problem frame for one requirement? We do not know. The entry in Table 3 shows the frame name, but questions whether this is worthwhile.

5.8. Email/SMS new trader

This interaction originates from the web machine and informs the customer of the successful activation of their new account, stating that they can commence trading. The outcome of the interaction is an email or SMS text message sent to the customer's mobile phone. Does this constitute a permanent domain of interest? Yes, since the message might be stored by the customer as a personal record. However, the customer is not required to store it, so it does not constitute a software development problem in that sense. What is a problem is the actual notification itself. The machine must generate a message and send it. The means of notification are dependent upon the details about the customer stored in the machine database. The requirement is only about the sending of the message (R12). A closer examination of the problem shows that the web machine must automatically notify the customer via email *and* by SMS if the customer record has a mobile phone number. How this is actually done would need further investigation. For an email message we can assume that the message is sent as a data

packet via a telephone communication network. For SMS, we might need an external party, an SMS generator, who is informed automatically via an email from the web machine that they should SMS the relevant customer phone with the message, CS. This would become a connection problem in Jackson's earlier discussion [16] and would look something like Fig. 14.

This is a simplification, of course, because the SMS generator would send the message via a series of mobile phone transmitters (cells) until the message reached the customer's phone, so there might be more connection domains between the web machine and the customer's phone. But this is out of our problem scope and we would not document it.

5.9. Inform bank

Despite the activation of an account by the web company, and that the customer can begin to trade or transfer funds online, the company still has to inform the high street bank of account activation. The bank should then open an account for the customer to store their trading/transferred funds. However, as the process model in Fig. A.1 shows, the bank conducts its own credit validation checks on the customer and can refuse to manage the customer's account. This appears to be a somewhat unusual arrangement. If the customer is rejected by the bank, the web company can still legally maintain the customer's account but would probably need further proof of the customer's solvency, such as recent utility bills. Since this had yet to occur when we were at the company (the bank had yet to reject a customer), it was uncertain what would really happen if the bank were to refuse (that's why the customer's state is documented as '? Alternative Active' in the process model). In any case, the interaction is the transference of the customer account details. Is there a persistent domain created? Yes, the customer account, an inert domain, but in this instance, it is in fact, a lexical representation of that domain since the state of the account will not alter as a result of the interaction and is therefore static. The interaction and outcomes meet the requirement that the bank be passed all activated customer account details via email, requirement R13. Is there a problem frame here in
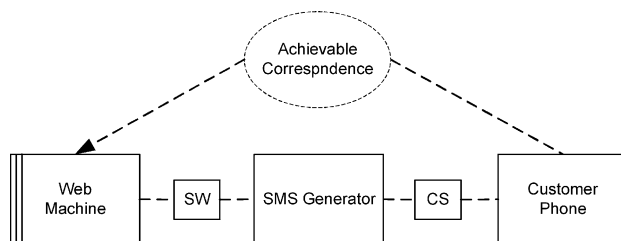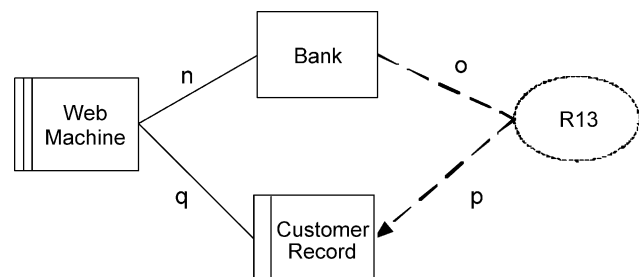

Fig. 14. Connection problem frame.

Fig. 15. Problem diagram for requirement 13.

the transference of the account details? There is nothing to control, no document to create/alter as in a workpiece and nothing to transform. But the bank does need the customer's particulars; this is an information notification problem as shown in Fig. 15.

The customer record is a realised domain and is constrained by the requirement p, to contain all of the information necessary to satisfy the bank, when it is received (o). The web machine retrieves the record data (q) and sends it to the bank (n). It is uncertain what type of frame this is. A realised domain would fit a transformation frame or a workpiece, but clearly this is not an exact fit. The problem is about information passing, something that is very relevant to the e-business domain. Does this then fit an information display frame? This too is unlikely since the concern is about notification, a transference of information, not how the bank displays it.

5.10. Acceptance message

This interaction is the combination of messages sent from the bank to the web machine either accepting or rejecting the customer account, and the subsequent passing of this message to the customer: four interactions are shown but only two occur in any one instance, either accept–accept or reject–reject. The outcomes of the interactions are either a data packet or email from the bank to the web system and email to the customer. These outcomes are static, since they are not manipulated or used in any way. There are no domains except the three roles in the process model. The requirement R14 stipulates that the company inform the customer of the bank's decision. This is a message-passing task that does not easily seem to fit one of Jackson's current problem frames [7] but does appear to be a connection problem [16]. Fig. 16 shows the problem frame.

Interface 'r' represents the message itself. We are making an assumption about this information and might have to reconsider whether the message is a domain of interest in its own right. The requirement 's' expects the bank to inform the web machine of the customer's status with the bank (r). The web machine itself is constrained to deliver the correct

message, 't'. The requirement 'u' states that the correct customer should receive the message.

6. Threats to validity

Our primary aim was to provide a solution to the industrial problem described in Section 2 so that the company could provide a faster service to their customers and make the print room staff jobs more efficient. The approach described in the paper arose from the circumstances of the project. The project was real. The end result was a released, updated software product. We conducted what amounts to action research whilst working as contracted engineers on this project. Our problem was a lack of understanding of the current software and application domain. It was therefore necessary to figure out what the requirements were. A sensible way to do this was to start with the simple process models developed by the business analyst and work from there. Problem frames were chosen as they propose a means of describing requirements problems and present a way to neatly decompose the larger problem into more manageable sub-problems. The fact that not everything worked perfectly with our proposed approach is part and parcel of real world software development [43]. But the fact that the company accepted our work and completed the project is validation enough. As one of the software developers stated, "[It is] good that everything is validated from the business [process] model through to design. We can see the clear progression from one phase to the next. We can specify what we want very clearly." The company's founder and vice-president positively commented, "Over-Overall [the] process [is] well worth considering for the IT department but it's more likely each department area will take what's useful to them and ignore the rest. Everyone can take little bits of this method for themselves and use that." The project manager stated, "[We] want…[the] work carried out [by the authors] to present as a full documentation set to the business." One developer commented that our approach is "requirements driven rather than data driven and that's not often how it works here. It would be better to apply the [authors'] method because specification is carried out and the IT department are wasting time trying to specify uncertain requirements" [39].

Discussion of sample is not entirely relevant in this case, since this was action research conducted on a real project. We were pragmatic in our approach since we had deadlines to meet. Validation came from company employees involved in the project. We do not claim the approach to be generalisable. We can only state what we found in this particular case though we do assert where we think there might be wider issues for the problem frames approach, as described in Section 5. There are also standard threats to
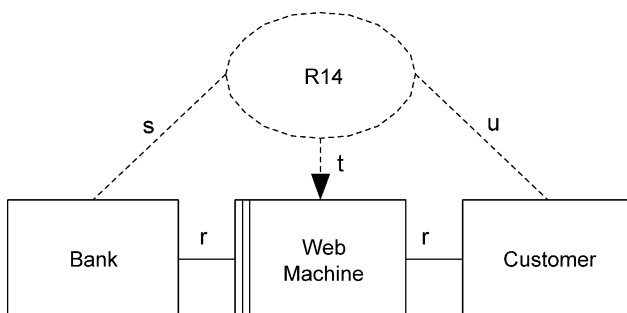

Fig. 16. Connection frame for requirement 14.

qualitative data validity: objectivity and reliability [44]. We address these threats below.

6.1. Objectivity

Objectivity concerns the biases that researchers might introduce into a study through their very presence or hidden agenda. We took on an action research role in the project since we had been asked to provide requirements engineering expertise by the company on a critical, live project. As such, we were deeply involved in achieving a successful outcome. The authors' intention for the project was not to test the proposed approach nor to evaluate problem frames, but rather to meet the project's objective of addressing a real business need: improving the application process for both potential customers and the print room staff team through a re-engineering of the existing product. As we view the project findings of importance to the software community, especially those interested in problem frames, we have provided an account of our findings here.

We show that there are some concerns with the problem frames approach (see Section 5), implying that further frameworks, methods or techniques will need to be devised to provide a starting point for problem domain oriented analysis, dependent upon the problem domain. We propose process modelling as one such potential starting point for the e-business domain.

6.2. Reliability

Since the qualitative researcher influences the research under study, a reasonable question to ask is, would someone else conducting the same study come to the same conclusions? This is impossible to say. It is well known that when conducting a software problem analysis, every analyst will emerge with a different solution, each potentially as equally valid as the next. However, given that there are a small number of domain types and a small number of problem frames it is entirely possible that the same problem frames will be discovered. However, for this to hold, the level of abstraction must first be understood and agreed. The process model in Fig. A.1 is envisioned, i.e. a process model for the proposed system rather than actual system. This means that actions within roles and inter-actions between roles might well be different if developed by another researcher, possibly leading to different problem frames. However, the process model and requirements elicited were in part derived from the existing process models and system, which provided a reliable validation mechanism. We validated the models with analysts, developers, managers involved on the project, and also the company vice-president. Validation meeting notes are numerous and can be found in [39].

7. Discussion

This study sought to investigate the application of problem frames within an industrial context. Partly to aid this process we chose to use process models as a way of helping to frame the problem. This was achieved by devising a simple set of stages in moving from process model to problem frame. However, the question must arise as to the validity of the approach. It was not always possible to elicit problem frames from the process models. Whether this is a fault with our approach, or indicates the more general difficulty of application of problem frames within our industrial context, is not clear.

Indeed, one might argue about the general validity of problem frames for e-business systems, and clearly more work on industrial application is required. The wealth of information gleaned that provides the problem context for the requirements is invaluable and can only be of benefit to the task of relating business process to its supporting systems. However, as stated, this is the only reported study that we know of which has examined the application of problem frames on a live industrial project. Certainly it would be useful to see how problem frames could be applied without use of process models. However, we argue that e-business developers are typically process focussed, and hence process modelling seems to be a natural first step.

Though much analysis has been done in this case study, and feedback from stakeholders suggests the use of the process models as a means of describing the product and providing traceability was most welcome, the problem frames approach was not accepted readily. The reason for this was, no doubt, unawareness of problem frames and doubts as to its usefulness in this particular problem. The company knew all about the domains and their properties. They understood what the problem was but needed it to be mapped out in a clear way:

> "It would be good to model every current system in IT with [the authors' approach] so we have details of everything—if we took this as a project in itself—and then every new project could expand on the model, rather than straight on the actual system."—IT development manager [39].

Though it might appear that we are suggesting that Jackson's five problem frames are not always appropriate to this particular e-business problem, this is in fact recognition of the utility of problem frames in general. As Jackson states, the problem frames approach is not a panacea [7]. This was never Jackson's intention, in fact, quite the opposite. The more generally applicable a method/tool/technique claims to be, the less likely it will help in any particular problem. We do not claim that the problem frames approach is entirely inappropriate to the e-business domain, either. We only

claim that we reached certain conditions where a recognised problem frame did not fit. This is not surprising when one considers the history of problem frames. The five frames emerged from traditional software engineering problems, such as control systems. We suspect that e-business problems are not always the same kind of problem as these.

8. Conclusions

This paper proposes a way to derive appropriate requirements from process models and couch them in terms of Jackson's problem frames. Therefore, we can consider the problem frames derived 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 constraints governing use or control of the domains). We identified two research questions in Section 1.5:

RQ1: Does the problem frames approach scale to complex, industrial projects?

Of the 14 requirements identified, only seven of Jackson's problem frames appear to fit. Two of these are transformation frames and are already solved by existing, standard software—and as such do not really need to be considered further. Two other frames are connection frames. However, Jackson does not consider the connection problem frame in his latest book [7], bringing into the question the validity of the identified frame, though they fit in our problem. Perhaps we need to reconsider the usefulness of the connection problem frame in the e-business domain. The two other frames are workpieces. One of the frames serves only one requirement—though there will be other occasions when the administrators will alter the contents of the customer's account, for instance, when there is a change of the customer's address. As such, this provides the workpiece with more persistence and we suggest this qualifies the frame as valid since it is only shown as one instance of usage of all the occasions that the company administrator may need to alter the state of the account.

On a number of occasions we derived problem diagrams but could not decide on the appropriate frame since one did not seem to fit. We therefore consider it quite possible that Jackson's frames are not always suitable to the e-business system domain. Indeed, we suggest that Jackson's frames have emerged from fairly traditional software engineering problems. The e-business domain has its own distinct characteristics: there is a lot of electronic message passing; the tools of e-business are standard hardware devices like servers and PCs. To couch a problem as an information display problem frame when we are referring to a PC

monitor as the display domain, seems a little too simplistic in this particular domain. We thus suggest that the key benefit of the problem frames approach is the approach itself, that is, as a mechanism for exploring and describing the problem domain and the requirements, not necessarily in the problem frames themselves. Critical to the future success of problem frames though, should be the ability to arrive at the appropriate problem frame that is suitable for the domain where one finds one's problem. The second research question was,

RQ2: What is a good starting point to begin a problem frames analysis in the context of the case study domain?

It is critical that one bounds the problem to its appropriate depth into the real world [7]. However, where that starting point is for e-business systems was uncertain, though recent work suggests business strategy as the right place to start for this domain [11–15]. Since the company described their problem boundary through simple process models, we chose our starting place through more complete process models, drawn as role activity diagrams. This provided a problem context and, hopefully for us, a prelude to problem frames. However, getting from a process model to a problem frame was not straightforward so we derived the approach described in this paper.

Though it is shown that although the approach appeared to show some promise [27] in its earliest phases and perhaps at its most abstract, a more detailed analysis reveals that not many problem frames can be gleaned from a number of interactions for the system studied. A means of improving our derivation method would be to add an extra step before 5. Once the requirements have been identified we should first describe the context for that requirement via a *problem diagram*. Only from there should we think about the appropriate problem frame, if we consider it will bring any added benefit to do so.

It is our intention to apply the approach to further industrial studies in the e-business domain. We also would like to then compare the problem frames approach with, for instance, a scenario-based approach, for deriving requirements, again in an industrial setting.

Acknowledgements

Appendix A
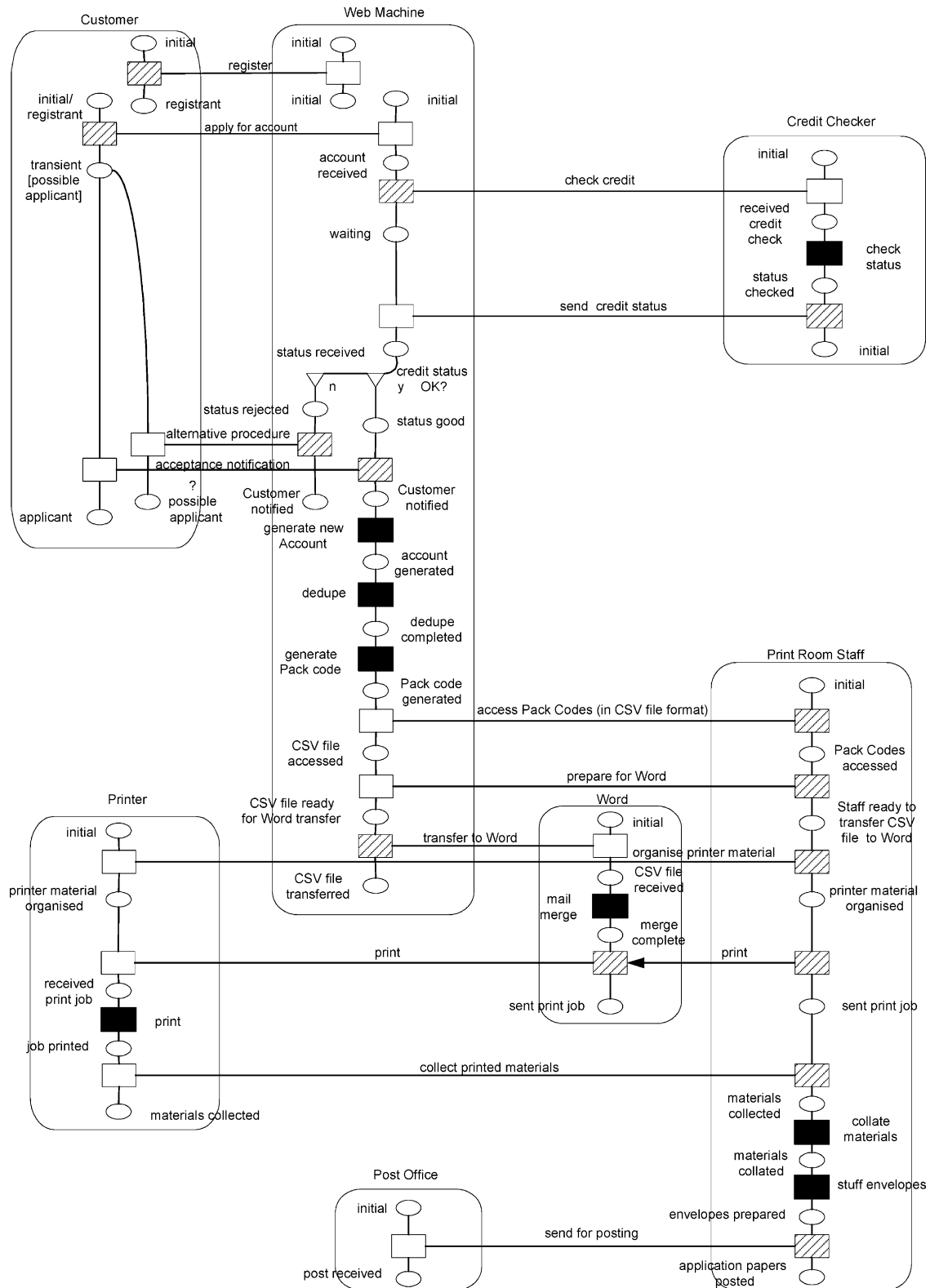
Fig. A.1.

Fig. A.1. Role activity diagram for envisioned product process.

Customer

applicant

send Trading forms/Bank forms to Customer

Post Office

post received

post sent

post received

sign application forms

forms signed

Return forms

Company Admin

initial

received forms

Update machine

Web machine updated

Web machine

initial

updated

activate account

account activated

email / SMS new Trader

email/SMS sent

waiting

Inform Bank of account details

Bank

initial

received A/C details

validate A/C

A/C validated

active

Bank informed

validation OK?

n   y

validation failed

validation successful

reject A/C message

message sent

message received

accept A/C

Inform Customer of Bank rejection

message received

? alternative active

Customer notified

message received

message sent

Inform Customer of Bank acceptance

message received

Customer notified

send account material to Customer

A/C material received
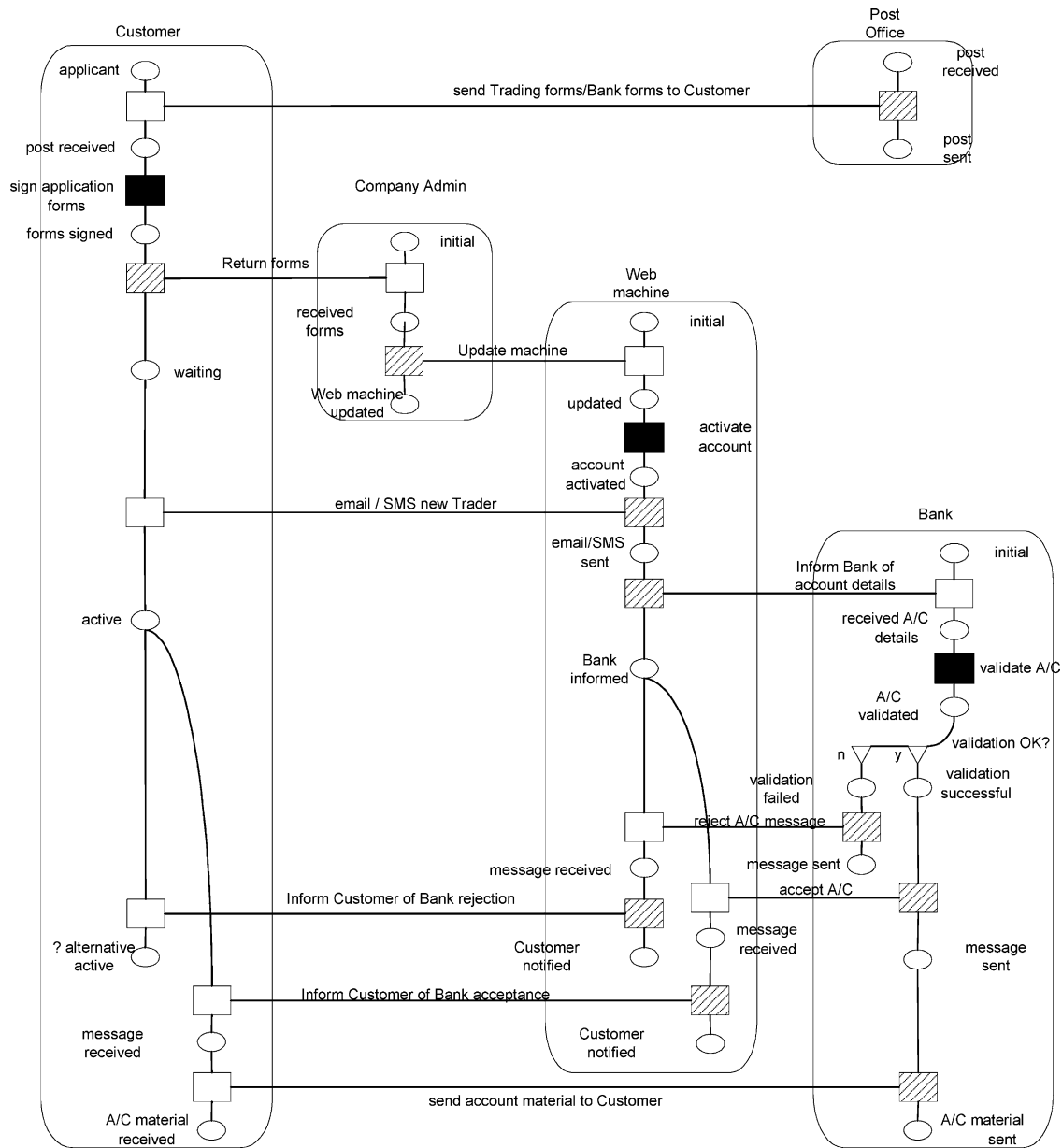
A/C material sent

Fig. A.1 (*continued*)

References

[1] P. Henderson, Software processes are business processes too, Third International Conference on the Software Process, IEEE Computer Society Press, Reston, VA, USA, October 1994, pp. 181–182.

[2] K.T. Phalp, The CAP framework for business process modelling, Information and Software Technology 40 (13) (1998) 731–744.

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

[4] G. Abeysinghe, K.T. Phalp, Combining process modelling methods, Information and Software Technology 39 (2) (1997) 107–124.

[5] K.T. Phalp, K. Cox, Guiding use case driven requirements and analysis, Seventh International Conference on Object-Oriented Information Systems, Springer, LNCS, Calgary, August 27–29, 2001, pp. 329–332.

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

[7] M. Jackson, Problem Frames, Addison-Wesley, Reading, MA, 2001.

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

[9] K. Phalp, K. Cox, Picking the right problem frame—an empirical study, Empirical Software Engineering Journal 5 (3) (2000) 215–228.

[10] P. Weill, M. Vitale, Place to Space: Moving to eBusiness Models, Harvard Business School Publishing, 2001.

[11] S. Bleistein, A. Aurum, K. Cox, P. Ray, Strategy-oriented alignment in requirements engineering: linking business strategy to requirements of e-business systems using the SOARE approach, Journal of Research and Practice in Information Technology 36 (4) (2004) 259–276.

[12] S. Bleistein, K. Cox, J. Verner, Problem frames approach for e-business systems, IWAAPF'04, First International Workshop on Advances and Applications of Problem Frames (an ICSE'04 workshop), Edinburgh, 24th May 2004, pp. 7–15.

[13] S. Bleistein, K. Cox, J. Verner, RE approach for e-business advantage, REFSQ'04, 10th International Workshop on Requirements Engineering: Foundation for Software Quality, Riga, Latvia, 7–8 June, 2004.

[14] S. Bleistein, K. Cox, J. Verner, Requirements engineering for e-business systems: integrating Jackson problem diagrams with goal modelling and BPM, APSEC 2004, 11th IEEE International Asia-Pacific Software Engineering Conference, Busan, S. Korea, 30th November–3rd December, 2004.

[15] S. Bleistein, K. Cox, J. Verner, Modelling business strategy in e-business requirements engineering, eCOMO'2004, Fifth International Workshop on Conceptual Modelling Approaches for e-Business, LNCS, Shanghai, China, 8–12th November 2004.

[16] M.A. Jackson, Software Requirements and Specifications, Addison-Wesley, Reading, MA, 1995.

[17] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, MA, 1995.

[18] I. Bray, An Introduction to Requirements Engineering, Addison-Wesley, Reading, MA, 2002.

[19] D. Bjorner, S. Koussoube, R. Noussi, G. Satchok, Michael Jackson's problem frames: towards methodological principles of selecting and applying formal software development techniques and tools, First IEEE International Conference on Formal Engineering Methods, IEEE Computer Society Press, Hiroshima, Japan, November 1997, pp. 263–270.

[20] J. Hall, M. Jackson, R. Laney, B. Nuseibeh, L. Rapanotti, Relating software requirements and architectures using problem frames, RE'02, 10th International Conference on Requirements Engineering, IEEE Computer Society Press, Essen, Germany, September 2002, pp. 137–144.

[21] L. Rapanotti, J. Hall, M. Jackson, B. Nuseibeh, Architecture-driven problem decomposition, RE'04, 12th International Conference on Requirements Engineering, Kyoto, Japan, September 6–10th 2004 (in press).

[22] K. Sikkel, R. Wieringa, R. Engmann, A case base for requirements engineering: problem categories and solution techniques in: A. Opdahl, K. Pohl, M. Rossi (Eds.), Sixth International Workshop on Requirements Engineering: Foundation for Software Quality, Stockholm, Essener Informatik Beitrage, Essen, 2000.

[23] M. Nelson, D. Cowan, P. Alencar, Geographic problem frames, Proceedings of the Fifth International Symposium on Requirements Engineering, Toronto, Canada, 27–31 August 2001, pp. 306–307.

[24] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, J. Moffett, Introducing abuse frames for analysing security requirements, 11th International Conference on Requirements Engineering (RE'03), Monterey, California, 8–12 September, pp. 371–372.

[25] I.K. Bray, K. Cox, The simulator: another elementary problem frame?, in: C. Salinesi, B. Regnell, E. Kamsties (Eds.), Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality—REFSQ 2003, Essener Informatik Beitrage, Velden, Austria, 2003, pp. 121–124.

[26] J. Tomayko, Adapting problem frames to extreme programming XP Universe Conference, Raleigh, NC ., 2001 http//www.xpuniverse. com/2001/pdfs/Edu02.pdf.

[27] K. Cox, K. Phalp, From process model to problem frame—a position paper in: C. Salinesi, B. Regnell, E. Kamsties (Eds.), Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality—REFSQ 2003, Essener Informatik Beitrage, Velden, Austria, 2003, pp. 113–116.

[28] H. Reubenstein, R. Waters, The requirements apprentice: automated assistance for requirements acquisition, IEEE Transactions on Software Engineering 17 (1991) 226–240.

[29] A. Sutcliffe, N. Maiden, The domain theory for requirements engineering, IEEE Transactions on Software Engineering 24 (1998) 174–196.

[30] N. Maiden, M. Hare, Problem domain categories in requirements engineering, International Journal on Human–Computer Interaction 49 (1998) 281–304.

[31] P. Coad, D. North, M. Mayfield, Object Models: Strategies, Patterns and Applications, Yourdon Press, 1995.

[32] M. Fowler, Analysis Patterns, Addison-Wesley, Reading, MA, 1996.

[33] S. Robertson, requirements patterns via events/use cases, http://www. systemsguild.com/GuildSite/SQR/Requirements_Patterns.html, viewed September 2003.

[34] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, Reading, MA, 2002.

[35] H. Kilov, Business Models: A Guide for Business and IT, Prentice-Hall, Englewood Cliffs, NJ, 2002.

[36] P. Weill, M. Vitale, What IT infrastructure capabilities are needed to implement e-business models, MIS Quarterly Executive 1 (1) (2002).

[37] M. Ould, Business Processes, Wiley, Chichester, 1995.

[38] K. Cox, J. Hall, L. Rapanotti (Eds.), Proceedings of the First International Workshop on Applications and Advances of Problem Frames—IWAAPF'04 (an ICSE'04 Workshop), IEE, Edinburgh, 2004.

[39] K. Cox, Heuristics for use case descriptions, PhD Thesis, Bournemouth University, 2002.

[40] E. Kamsties, K. Hormann, M. Schlich, Requirements engineering in small and medium enterprises, Requirements Engineering Journal 3 (1998) 84–90.

[41] C. Britton, J. Doake, Software System Development: A Gentle Introduction, McGraw-Hill, 1993.

[42] D. Garlan, Problem types and problem frames, Lecture, http://www-2. cs.cmu.edu/afs/cs.cmu.edu/project/tinker-arch/www/html/1998/Lectures/03.ProbFrames/base.000.html.

[43] A. Davis, A. Hickey, Requirements researchers: do we practice what we preach?, Requirements Engineering Journal 7 (2002) 107–111.

[44] M. Denscombe, The Good Research Guide, Open University Press, 1998.