# Towards process design for efficient organisational problem solving

## Conference or Workshop Item

# Towards Process Design for Efficient Organisational Problem Solving

Dariusz W. Kaminski
Computing Department
The Open University, UK
dariusz.kaminski@gmail.com

Jon G. Hall
Computing Department
The Open University, UK
J.G.Hall@open.ac.uk

Lucia Rapanotti
Computing Department
The Open University, UK
L.Rapanotti@open.ac.uk

***Abstract*—This paper presents initial results on stochastic business process modelling. We interpret business processes as problem solving processes to show that certain trade-offs in stakeholder involvement can usefully be analysed. Our initial results illustrate the analytic potential of our stochastic model, which could be useful to business process analysts and designers.**

***Keywords*-process modelling; business process design; Problem Oriented Engineering**

## I. Introduction

Organisational problem solving has become an incredibly complex topic [4]. The development of business processes as an organisation's response to problems they face is hampered by this complexity, and the tools currently available for solving organisational problems are relatively unsystematic.

Research by the second and third authors in Problem Oriented Engineering (POE) offers an approach to the structuring of organisational problems that has been successfully applied in many organisational contexts, from the design of seating arrangements to business process reengineering in a financial engineering setting.

In this paper, we interpret part of POE as a stochastic process and consider the analytic potential this provides for the modelling of business processes.

The paper is organised as follows. Section II-A provides some background on POE. Section III presents the link between POE and business processes. In Section IV, we map POE Process Pattern to its stochastic representation. Section V gives an overview of the study from the industrial context, which we then model with our stochastic model. An evaluation of the study and its results is given in Section VIII. Related work is discussed in Section VI, while Section VII concludes the paper.

## II. Background

### A. Problem Oriented Engineering

Rogers' definition of engineering [7] states that[1]:

> "Engineering refers to the practice of organising the design and construction of any artifice ($S$) which transforms the physical world ($E$) around us to meet some recognised need ($N$)."

Problem Oriented Engineering encodes this definition as a problem solving exercise – an engineer's task is to find $S$ that satisfies $N$ in environment $E$: represented by the proposition $E, S \vdash N$ meaning that that, when $S$ is installed in the environment $E$, their combination meets the need $N$[2].

POE characterises the problem solving process as illustrated in Figure 1. Briefly, in the POE Process Pattern (PPP) there are four groups of agents that interact with each other during the problem solving activity: problem explorers, problem validators, solution explorers and solution validators. During problem (resp., solution) exploration a problem's context and requirements (resp., a solution) are understood and described. Problem (resp., solution) validators are available to problem (reps., solution) explorers and will perform validation (if requested) of the current problem (solution) description [2].

[1] Abbreviations $E$, $S$, and $N$ added by the authors.

[2] For reasons of space, we do not expand on the usage of POE problem notation; the interested reader is referred to [3].

Each state in the PPP can be linked to the state of a problem's elements during problem solving, states represented by $\times$ (meaning element unknown or invalid), $c$ (meaning element has a candidate description) and ✓ (meaning element has been validated). Thus, for example, $E\checkmark, Sc \vdash N\checkmark$ represents a POE problem whose environment and need have been validated (by a problem validation step), and a solution (candidate) has been proposed but not yet validated.

As shown, unsuccessful problem validation will typically result in problem exploration being extended. Successful problem validation will typically result in solution exploration being begun. Problem validation allows developmental risk to be transferred from explorer to validator. Unsuccessful solution validation is more complex as not only can it reveal that the solution does not satisfy a problem (the feedback loop between solution validation and problem exploration in Figure 1), but also that the problem was misunderstood (whether validated or not; the arc to 8 in Figure 1).

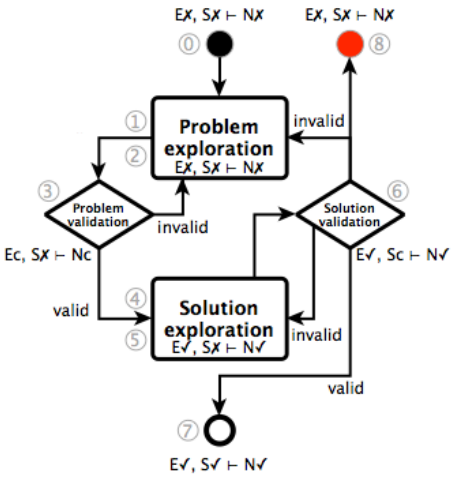


Figure 1. POE problem $E, S \vdash N$ at different points of problem solving in PPP. The numbers representing states will be explained in Section IV.

## III. Business Processes as Problem Solving

POE has been shown capable of modelling business processes (for instance, [6]). As a result of that modelling, effective business process changes have been suggested based on the properties that POE processes are expected to have. One could argue, however, that this success is to some extent fortuitous, requiring business processes that are susceptible to modelling in POE and it is not obvious that all business processes can be seen in this way. So, to be able to speak about general business processes in a POE setting, we must be able to encode them as problem solving processes.

Business processes are defined by van der Aalst and Stahl [9] as:

> "A business process consists of a set of activities that is performed in an organisational and technical environment. These activities are coordinated to jointly realise a business goal."

and so can be associated with a) an organisational and technical environment, and b) a business goal. To show that a business process is a problem-solving process in the POE sense, we will construct a problem that the business process solves from this environment and goal (the need that the solution satisfies). For the business process $BP$, call the environment $ENV_{BP}$ and the goal $GOAL_{BP}$. Then form the problem:

$$Problem_{BP} : ENV_{BP}, S \vdash EstablishGOAL_{BP}$$

We see that:

1 $Problem_{BP}$ is a POE problem;
2 $BP$ solves $Problem_{BP}$ in that it delivers $S$.

Hence, to each business problem we can associate a POE problem that it solves. In essence, from an engineering perspective, business processes are designed and implemented in response to organisational problems. The above construction shows that business processes are problem solving processes, and we have given a representation of them in POE.

## IV. Stochastic semantics

In this section we model a single instance of the PPP as a Markov chain, where we map PPP

elements to states (numbers in Figure 1). The PPP process can be seen as a transition system. While the choices made by the validators will not be known until the execution, we can model the behaviour with Markov chains. They are a useful means of modelling processes, and can be seen as transition systems, in which state transitions are decided probabilistically [1]. This extends the PPP model by a set of probabilities, which could be interpreted as characteristics of the agents taking part in the problem solving.

By mapping the POE Process Pattern to a Markov chain, and for the purposes of PRISM encoding, we identified nine states (state *p* in Figure 1 and Figure 2 – numbered from (0) to (9)). States (0) and (7) represent the process in the initial and successfully solved state (they correspond to the black and white circles in Figure 1, respectively). Problem exploration (2) and solution exploration (5) are followed by their respective validation check points: problem validation (3) and solution validation (6). Additionally, we introduced states (1) and (4) to model resource expenditure during explorations, i.e., transitions *Pexp* and *Sexp* increment the cost. When the process runs out of resources, or when the solution validator declares it unsolvable, then it remains unsolved – state (8). All these states, and their transition probabilities are shown in Figure 2.

With our model of PPP encoded as a Markov chain, we can use PRISM model checking tool [5], which allows for process simulation and analysis. It is useful, because with this we can model the overall probability of successful problem solving under budgetary constraints, i.e., we will be able to answer questions such as, for example, this: *with a given budget of X, what is the probability that the problem solving will be successful?*

In Figure 3, the Markov chain is based on a set of probabilities guarding the transitions (lines from 2 to 8), and it is composed of two modules: `PPP` and `Cost`. The former represents the states and transitions between. For example, in line 16 we have an action `[PASK]` from state `p=2` to states `p=3` or `p=4`, guarded by their respective probabilities *Pask* and $1 - Pask$. Module `Cost` declares two actions

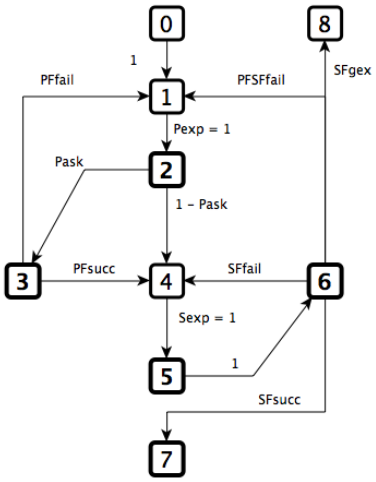


Figure 2. PPP as Markov chain.

`[PEXP]` and `[SEXP]`, each of which increments variable `cost` by an arbitrary value of 1. Formula `Res` is a guarding condition for many transitions of the model, which monitors if the overall cost has reached the budgetary constraints (integer variable `BUDGET`).

### A. Problem solving resource usage

Assigning arbitrary costs to each exploration action in the DTMC allows us to consider the overall cost of problem solving as probabilities on the arcs change. Each team expends resources, and when a process runs out of resources, it declares the problem as unsolved.

Figure 4 shows the results from running experiments on our PPP model in PRISM, with discrete `BUDGET` (in this case, values from 0 to 10 are representative enough, curves for values closer to 10 will be similar, i.e., the more budget available, the higher the probability of success), where we plot the curves representing the probability of successful solving for different values of *PFsucc*. The process never completes when $BUDGET < 2$ (the lines for *BUDGET* equal to 0 and 1 lie on the X-axis), because PPP model needs at least 2 units of resource to complete both explorations. Depending on the available budget, we can state that with more available budget the probability of

```
dtmc
        const double PFfail = 1 - PFsucc;
        const double PFsucc;
        const double Pask = 1.0;
        const double PFSFfail = 0.1;
        const double SFfail = 0.1;
        const double SFsucc = 0.7;
        const double SFgex = 0.1;
        const int BUDGET;
        const int Unsolved = 8;
        formula Res = (cost > BUDGET);
module PPP
        p: [0..8];
        [START] p=0 & !Res -> 1: (p'=1);
        [PEXP] p=1 & !Res -> 1: (p'=2);
        [PASK] p=2 & !Res -> Pask: (p'=3) + (1-Pask): (p'=4);
        [PFVAL] p=3 & !Res -> PFfail: (p'=1) + PFsucc: (p'=4);
        [SEXP] p=4 & !Res -> 1: (p'=5);
        [SASK] p=5 & !Res -> 1: (p'=6);
        [SFVAL] p=6 & !Res -> PFSFfail:(p'=1)
                + SFfail:(p'=4)
                + SFsucc:(p'=7) + SFgex:(p'=8);
        [UNSOLVED] Res -> 1: (p' = Unsolved);
        [SOLVED] (p=7) -> 1: (p'=7);
        [GEX] (p=8) -> 1: (p'=8);
endmodule
module Cost
        cost : [0..BUDGET];
        [PEXP] !(cost >= BUDGET) -> 1: (cost'=cost + 1);
        [SEXP] !(cost >= BUDGET) -> 1: (cost'=cost + 10);
endmodule
```

Figure 3. PPP as a discrete-time Markov chain `dtmc` model in PRISM, which comprises two modules: `module PPP` and `module Cost`.

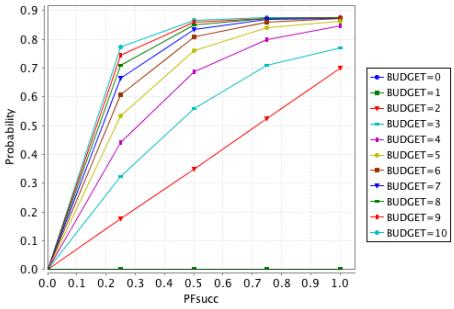


Figure 4. Probability of successful problem solving depending on *PFsucc* (curves plotted for different values of the available `BUDGET`).

success increases. We can see that the better the team, the higher the overall probability of success.

In Figure 5, we plotted curves for probability of successful solving, and we observe, that this probability is increasing for increasing values of *BUDGET*. Since we only consider discrete values of *Budget*, from the plotted points we see, that probability of solving is 0, when $BUDGET < 2$.

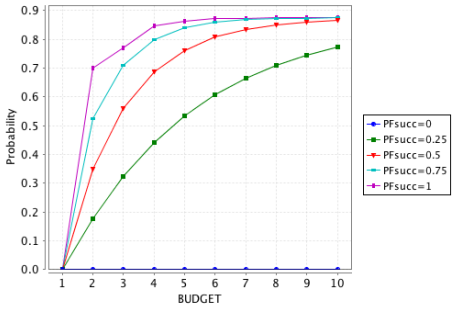


Figure 5. Probability of successful problem solving depending on the available `BUDGET` (curves plotted for different values of probability *PFsucc*).

These results confirm that the probability of success depends on *PFsucc* – quantitative characteristic of our problem solving team, and in this case, the better the team (and the more *BUDGET* available), the higher the likelihood of solving the overall process.

### B. Team characteristics

It is reasonable to characterise the problem and solution exploration expertise of agents that perform them. A good problem exploration team is likely to reach a validatable problem understanding more quickly and within budget. Conversely, a poor team can spend all allocated resources without producing a validatable problem understanding – leaving a problem unsolved. In the Markov model, poor problem exploration team expertise translates to high *PFfail* ($= 1 - PFsucc$) and *vice versa*. Similarly, poor solution exploration team expertise translates to low *SFsucc*. Other probabilities in the model relate to: the teams' propensity to ask for validation (*Pask*), probability of failed problem validation (*PFfail*), probability of failed solution validation due to an invalid problem (*PFSFfail*), probability of failed solution validation (*SFfail*), and finally, probability of a catastrophic solution validation leading to a 'global' exception (*SFgex*). We should note that, as for any Markov chain, the sum of transition probabilities from any given state has to be equal to 1, e.g., for state (6), we have $SFsucc + SFfail + PSSFfail + SFgex = 1$.

## V. Case study

Nkwocha et al. model part of the defect tracking process of a mortgage calculation software [6]. Essentially[3], software defects lead to incorrect mortgage calculations. There are two remedies, the first is tactical – wrong data values are corrected and the calculation retried; the second is strategic – the software is debugged and the session rerun.

Tactical solutions are less expensive than strategic solutions and so are preferred by the solution provider's perspective. However, a tactical solution does not always solve the problem and, when it fails, the customer has to request the strategic solution leading to customer satisfaction. From the customer's perspective, it is better to choose to implement tactical or strategic based on problem analysis.

In this section, we model the trade-off between the solution provider's and the customer's positions.

The process is complex (see [6]) so, for reasons of brevity, we have modelled before and after as shown in Figure 6 in which a) shows the process without validation and b) with problem validation. Again for simplicity, and rather than build DTMC two models, we can use the probabilities associated with the state 2 in Figure 3 to omit problem validation ($Pask = 0$) and include it ($Pask = 1$).

We set up a simple PRISM-based experiment based on the DTMC shown in Figure 3. We varied *Pask* between 0 and 1 in increments of 0.2 and measured the probability of a successful solution being found ($P(succ)$) for a given budget for each datum. The results are shown in Figure 5 and described below.

For the before case of wholly tactical solutions (modelled by $Pask = 0$, the black line in Figure 5), the reader will note that $P(succ)$ increases slowly with available *BUDGET*. For example, when the process has 10 units of *BUDGET* to spend, $P(succ) = 0.4$ (Figure 6a). For the after case of problem analysis with customer validation, (modelled by $Pask = 1$, the green line in Figure 5)there is sharp rise from $P(succ) = 0$, i.e., a solution is never found to $P(succ) = 0.95$, a solution is almost always found. If the available budget is less than the cost of strategic solving (here, 10 units), it is better not to ask for PV, i.e., if $Pask > 0$, then PV could suggest expensive (over budget) solution, which would reduce the probability of successful solving.

The other values of *Pask* in Figure 5 are shown to illustrate the nature of the discontinuity around a *BUDGET* of 10.

[3]Simplifying for reasons of space; full details can be found in [6].

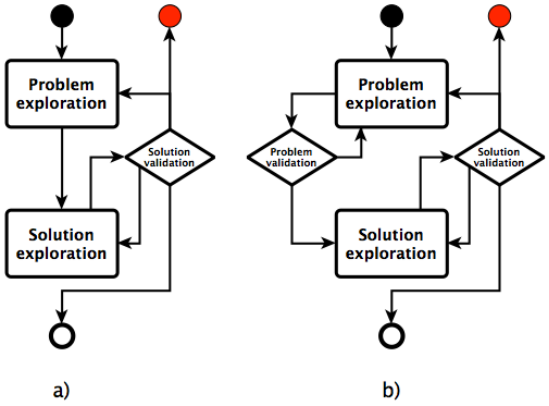


Figure 6. Process a) without asking for problem validation ($Pask = 0$); b) with problem validation ($Pask = 1$).

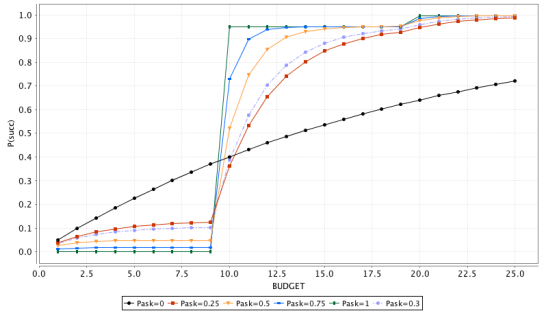


Figure 7. Results of PRISM-based experiment: the relationship between the available *BUDGET* and the probability of successful solving $P(succ)$ for various values of *Pask*.

## VI. Related Work

With the abundance of business process modelling techniques, many of them are centred around capturing and visualisation, and only a limited

number of techniques allow for quantitative analysis and structured process improvement [10]. While the business process modelling domain has become a ubiquitous part of the modern business enterprise, and most organisations view their operations in terms of processes [8], many approaches suffer from the common issues related to choice of languages, standardisation, and interoperability. Our approach of mapping POE processes to stochastic models goes beyond a visual process representation, because it makes it possible to annotate states and arcs with characteristics, which could then be modelled, analysed and simulated.

A good example of an approach to business process modelling that proved to be stable and relevant are Petri Nets [9]. They combine both visual and formal aspects, and can be used for quantitative modelling of processes. Our approach to modelling of POE processes is based on the POE Process Pattern, which unlike Petri Nets, has no explicit notion of the flow. Instead, the focus is on activities performed by agents, and the exchange of information between them in order to transform a POE problem $E\times, S\times \vdash N\times$ into $E\checkmark, S\checkmark \vdash N\checkmark$.

## VII. Discussion and Conclusions

## VIII. Discussion

We have characterised business processes as problem solving activities, and used this result to model them as stochastic processes using the POE Process Pattern. This has allowed us to develop a model in which we measure the comparative probabilities of success under simple process transformations: *viz.*, with and without problem validation. We have applied our techniques to a case study from [6], which provides supporting evidence that the transformation described there lead to more cost effective processes. However, we have also identified that there is a point before which the trade-offs are not economic. This is a new result and one that we will further investigate.

Based on the stochastic semantics presented here, future research will explore characteristics of more complex models, i.e., we will investigate arbitrarily complex processes created by composing POE processes in sequence, parallel and fractally [**?**].

## References


[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[2] J. G. Hall and L. Rapanotti. Assurance-driven design 2.0. Technical Report TR2009/03, Computing Department, The Open University, 2009.

[3] J. G. Hall, L. Rapanotti, and M. Jackson. Problem oriented software engineering: A design-theoretic framework for software engineering. In *Proceedings of 5th IEEE International Conference on Software Engineering and Formal Methods*, pages 15–24. IEEE Computer Society Press, 2007. doi:10.1109/SEFM.2007.29.

[4] M. Hobday, G. Dosi, and L. Marengo. Problem solving behaviours, organizational forms and the complexity of tasks. In C. Helfat, editor, *The SMS Blackwell Handbook of Organizational Capabilities*, pages 167–192. Blackwell, 2003.

[5] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[6] A. Nkwocha, J. G. Hall, and L. Rapanotti. Design rationale capture in the globalised enterprise: An industrial study. In *Proceedings of Fifth International Conference on Software Engineering Advances (ICSEA 2010)*. IEEE, 2010. Electronic proceedings.

[7] G. Rogers. *The Nature of Engineering*. The Macmillan Press Ltd., 1983.

[8] A. ter Hofstede and N. Russell. *Modern Business Process Automation*, chapter The Language: Rationale and Fundamentals. Springer, 2010.

[9] W. van der Aalst and C. Stahl. *Modeling Business Processes: A Petri Net-Oriented Approach*. The MIT Press, 2011.

[10] K. Vergidis, A. Tiwari, and B. Majeed. Business process analysis and optimization: beyond reengineering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(1):69–82, 2008.