

Configuring Service-Oriented Systems using PEPA and AI Planning

Amanda Coles, Andrew Coles

Department of Computer and Information Sciences,
University of Strathclyde, Glasgow, UK
firstname.lastname@cis.strath.ac.uk

Stephen Gilmore

School of Informatics,
University of Edinburgh, UK
firstname.lastname@ed.ac.uk

September 1, 2009

Abstract

In this paper, we look at how two hitherto-distinct approaches to system modelling — PEPA and AI Planning — can be combined to provide configuration decisions for service-oriented systems. By combining the modelling of uncertainty in PEPA with the fast decision making of planning, we exploit their respective strengths. To illustrate the potential of our approach, we consider a model of a system for obtaining loan decisions from a broker, for which we produce cost-effective investment decisions to meet a desired performance target.

1 Introduction

The cost-effective provision of Service-Oriented Systems — ‘systems of systems’ set up to provide a desired service — presents an interesting challenge to 21st century computing. Considering two facets of the problem, first, there is the challenge of modelling expected system performance given the performance of system components and network infrastructure. Second, there is the issue of choosing the system configuration: the possibility of distributing data processing on a global scale gives considerable choice over how to compose the service, and as such finding the most cost effective system configuration to meet a desired performance target is non-trivial. For instance, a cheaper remote service is only cost-effective if the investment in bandwidth needed to meet the desired performance is reasonable; but at the same time, investing elsewhere in the system may make investment in bandwidth unnecessary.

In this paper, we explore how two approaches to system modelling can be combined to meet these challenges. For an accurate performance model, we will use PEPA [4]. For fast decision making, we will use AI Planning [2], working with an approximate model of system performance. The two systems, combined, form a closed loop: when the model of performance provided initial PEPA model is unacceptable, planning is used to provide cost-effective investment decisions; then, an updated PEPA model is built to ascertain whether the performance target has now truly been met.

2 Background: AI Planning

AI Planning is concerned with the task of finding a sequence of actions that, when executed, reach some desired goals. Beginning with planning in its simplest form — propositional planning, without an explicit notion of time or numbers — a planning problem can be described by a tuple $\langle I, G, A \rangle$, where:

- I is the *initial state*: a set of propositions that hold true initially (under the closed world assumption that any fact not in I is initially false);
- G is the *goal state*: a set of propositions such that $G \subseteq S_g$ for any goal state S_g ;
- A is the set of *actions*, with the aim being to reach a state S_g through applying successive actions from A to I .

Each action $a \in A$ can, in turn, be defined by a tuple $\langle pre, del, add \rangle$, where:

- pre is the *precondition* set of a : for a to be applied in a state S , $pre \subseteq S$;
- del and add are the set of facts deleted (resp. added) upon the application of a .

```

(:action drive-truck
:parameters (?t - truck ?from ?to - locations)
:precondition (and (at ?t ?from)
                   (>= (fuel ?t) (/ (distance ?from ?to) (mpg ?t))))
)
:effect
  (and (not (at ?t ?from))
        (at ?t ?to)
        (decrease (fuel-in ?t) (/ (distance ?from ?to) (mpg ?t))))
)
)

```

Figure 1: Moving a truck

Applying a in a state S then gives a state S' where:

$$S' = (S \setminus del(A)) \cup add(A)$$

More recently [3], planning models written in the Planning Domain Definition Language (PDDL) can include numeric state values in their specification. An example of an action manipulating numeric values is shown in Figure 1. As can be seen, for the action to be applied, the truck must be at the designated start location, and hold enough fuel for the journey; and when the action is applied, the location of the truck is updated, and the amount of fuel in it decreased accordingly. With models such as these, the task of a modern planner such as COLIN [2] is to find a solution plan that respects this defined behaviour, along with other capabilities of PDDL, such as modelling time and continuous numeric processes (e.g. filling a tank at a certain rate).

3 Modelling Service-Oriented Systems in PEPA

We will use PEPA [4] to model a service-oriented system where a customer (CR) requests loans from a broker (BR), who forwards on the request to a lender (LE), and returns the response to the customer. The customer, broker and lender are physically distributed and communicate across a network between the customer and the broker (CB) and a network between the broker and the lender (BL).

The Service-Level Agreement (SLA) for this system is expressed in terms of the response time experienced by the customer. That is, the delay between the end of the request activity and the end of the response activity. (It does not matter how long the customer takes to make a request, we are concerned only with how responsive is the service composition of broker and lender, once the request has come in.) The SLA of the loan service states that 80% of customers receive a response within 7 seconds. The SLA does not state what percentage of these responses turn down the loan request.

The customer (CE) makes a request and then is ready to receive a loan offer. When this is completed, the customer thinks about the offer, and the process repeats.

$$\begin{aligned}
CR &\stackrel{def}{=} (request, r).CR_{ready} \\
CR_{ready} &\stackrel{def}{=} (response, \top).CR_{completed} \\
CR_{completed} &\stackrel{def}{=} (think, t).CR
\end{aligned}$$

The network between the customer and the broker is symmetric in the sense that it is reasonable to use only a single rate parameter r_{CB} to represent transfer from customer to broker in both directions (first for the request, then for the response).

$$\begin{aligned}
CB &\stackrel{def}{=} (request, \top).(transferToBroker, r_{CB}). \\
&\quad (transferFromBroker, \top).(response, r_{CB}).CB
\end{aligned}$$

The broker (BR) is idle until a loan request comes in. The broker then identifies a suitable lender by consulting a registry and then forwards the loan request to the lender. (The interaction with the registry is not represented here.) When the response is received from the lender the broker post-processes it and transfers it back across the network to the customer.

$$\begin{aligned}
BR &\stackrel{def}{=} (transferToBroker, \top).(requestLender, r_{BR}). \\
&\quad (responseLender, \top).(transferFromBroker, r_{BR}).BR
\end{aligned}$$

The network between the broker and the lender is also symmetric, imposing the same delay when transferring the request to the lender, as when transferring the response from the lender. Thus the rate r_{BL} is used twice in the definition of this

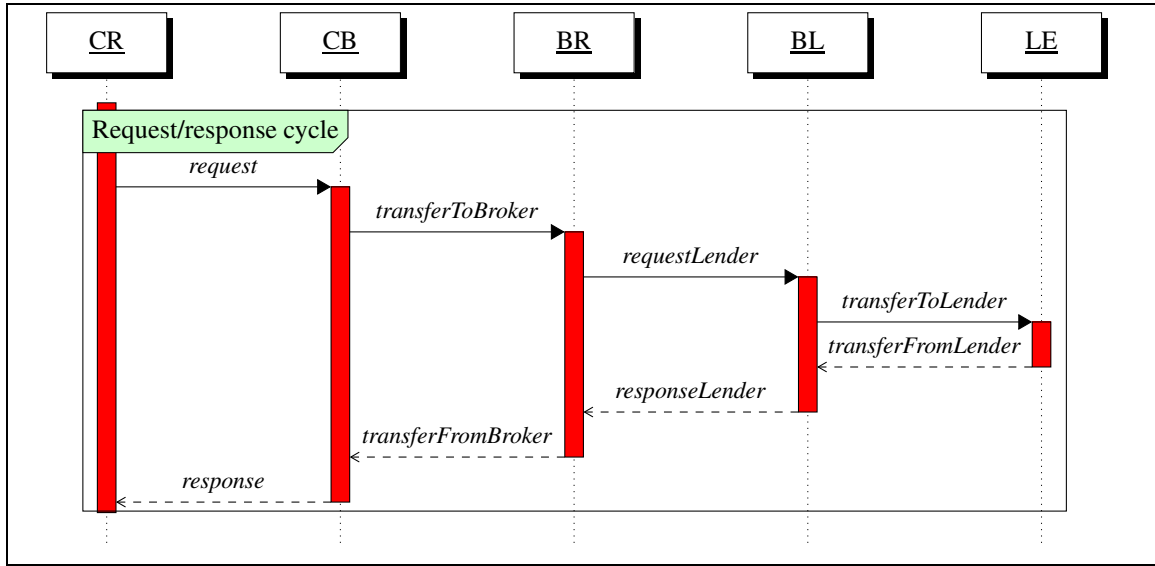


Figure 2: A UML sequence diagram showing the interactions between the customer (CB), the broker (BL) and the lender (LE).

component.

$$BL \stackrel{\text{def}}{=} (requestLender, \top).(transferToLender, r_{BL}).BL \\ (transferFromLender, \top).(responseLender, r_{BL}).BL$$

The lender (LE) simply functions as a reactive system, being inactive until receiving a request and then giving back the loan offer.

$$LE \stackrel{\text{def}}{=} (transferToLender, \top).(transferFromLender, r_{LE}).LE$$

Finally, the complete PEPA model is a composition of these five sequential components, requiring them to cooperate on their common actions.

$$(CR \bowtie_{\mathcal{L}_1} (CB \bowtie_{\mathcal{L}_2} (BR \bowtie_{\mathcal{L}_3} (BL \bowtie_{\mathcal{L}_4} LE))))$$

where

$$\begin{aligned} \mathcal{L}_1 &= \{ request, response \} \\ \mathcal{L}_2 &= \{ transferToBroker, transferFromBroker \} \\ \mathcal{L}_3 &= \{ requestLender, responseLender \} \\ \mathcal{L}_4 &= \{ transferToLender, transferFromLender \} \end{aligned}$$

The activity relevant to the SLA is depicted as a UML diagram in Figure 2. The five sequential components of the PEPA model are represented as swimlanes which proceed down the page. The horizontal lines which cross from one swimlane to another mark the end of an activity. The height of a thick bar represents the duration of one or more activities. We are concerned with the height of the bar in the CB swimlane, which represents the seven-activity sequence $transferToBroker$; $requestLender$; $transferToLender$; $transferFromLender$; $responseLender$; $transferFromBroker$; and $response$. These activities need to take place (in this order) to complete a request from a customer.

4 Approaching PEPA Models from Planning

As discussed in the previous section, PEPA provides an elegant means whereby service-oriented systems can be formally defined. Its key strength in this setting lies in the ability to use distributions to model the how long each activity takes, capturing the uncertainty in the performance of each of the system's components. In doing so, it is possible to verify service-level agreements (SLA) specifying a total service time and the percentage of transactions for this must hold. The weakness of the approach is encountered when the service-level agreement is not met, and hence where investment is needed to reduce one or more activities' durations. There is no direct way to consider these possibilities: the modeller is left with the task of performing sensitivity analyses on the system to deduce where investment should be applied, whilst bearing in mind the costs of doing so.

The strengths and weaknesses of planning models contrast strongly with that of PEPA in this domain. As discussed in Section 2, a PDDL model can capture a sequence of activity, but unlike PEPA, in classical planning the outcome of the

```

(:action CB1
:parameters ()
:precondition
  (ready)
:effect (and
  (not (ready))
  (ready_for br1)
  (increase (dur) 0.3)
)
)

(:action BR1
:parameters ()
:precondition
  (ready_for br1)
:effect (and
  (not (ready_for br1)))
  (ready_for br1)
  (increase (dur) 0.9)
)
)

```

Figure 3: CB1 and BR1 Actions from the Basic PDDL model

action is deterministic: moving the truck uses precisely the calculated amount of fuel, and were a temporal model used, it would take precisely the calculated amount of time. Where planning has its strengths is in considering the implications of decisions and how well they contribute towards the goal. Actions can be used to encode the options available and their costs, and the planner can then consider these when planning to reach the goals. For instance, in the case of a logistics problem, the costs would correspond to refueling the trucks and employing drivers.

From these comparisons, what we seek is useful middle ground combining the strengths of these two approaches. Where a PEPA model falls short of the SLA, investment decisions are needed — a strength of planning. But, to make useful investment decisions, the planner needs to consider uncertainty — a strength of PEPA models. Here, we shall proceed to answer the following: how can we give a planner enough information about the uncertainty in activities’ durations to allow it to make cost-effective service investment decisions, leading to a modified PEPA model which is then feasible? The question is somewhat different in character to previous work on considering uncertainty in planning, as the outcome of search is a plan of investment which configures a system capable of meeting the SLA under the modelled uncertainty, rather than a policy to guide the transaction through the system whichever of the uncertain outcomes happens.

4.1 A Basic PDDL System Model

First, we shall construct a basic PDDL model corresponding to the steps of the GetLoan system (Figure 2). Our initial state declares that the system is idle (the fact `ready` holds), and the goals are that the second `CB` activity has finished, and the total time taken is ≤ 6 seconds. For each activity, a corresponding action is created in the planning problem; for activities in two fragments (e.g. `CB`) we add two actions, `CB1` and `CB2`. Two example actions are shown in Figure 3. First, considering numeric effects, each action increases the duration of the activity sequence by the mean time taken to complete the activity. Second, considering the propositional preconditions and effects, the first of these adds a fact required as a precondition for the second. Similar effect–precondition constructs exist for the remaining actions, until `CB2` adds `done`, as required by the goal state.

4.2 Modelling Uncertainty and Choice

The PDDL model, as described, only considers the mean time taken to complete each activity, taking the total time to complete the sequence as the sum of these. As there is no model of uncertainty, the planner has no basis on which to determine the distribution of possible outcome durations, and hence no means to determine whether the SLA will hold under the exponential distributions used as the basis for the original PEPA model. To address this, we shall approximate these distributions within the PDDL model, allowing the planner to estimate the attainable SLA.

To achieve this, we make two changes to our model. First, each action has a number of duration outcomes, associated with likelihoods, rather than a single duration effect. For durations samples d_0, \dots, d_n taken from the time distribution for activity a , the corresponding likelihood $p_i \in p_0, \dots, p_n$ will be taken as $(CDF(d_i, \lambda_a) - CDF(d_{i-1}, \lambda_a))$ (defining $CDF(d_{-1}, \lambda_a) = 0$). Second, each action applies its outcomes onto each of the previous possible outcomes. In doing so, once the plan contains all the necessary activities, by summing the likelihood of all outcomes within the target deadline (e.g. 6 seconds), we can see if the target likelihood (e.g. 90%) has been met.

Before illustrating this with an example, we make one final change to the model to reflect choice over the investment decisions available to improve the system’s performance. For each activity, a range of rates are available, with associated costs. By planning to minimising the sum of these costs, whilst also meeting the SLA, the planner can answer questions of the form ‘what is the minimum investment needed to meet this SLA?’. Figure 4 shows an example of the finished action model. The cost effect is fairly intuitive — increase the total cost by the cost of setting `BR1` to follow the rate chosen. Also note the effect `(rate_for_br2 ?r)`, which records that this rate must later be used for `BR2`.

The outcome effects are slightly more involved. At a superficial level, whenever an action is applied, we update the space of possible outcome durations and likelihoods for the plan so far (as a whole) to reflect the space of possible

```

(:action BR1
:parameters (?r - rate)
:precondition
  (ready_for br1)
:effect (and
  (not (ready_for br1))
  (ready_for bl1)
  (increase (cost) (cost_br1 ?r))
  (ratefor_br2 ?r)
  (forall (?sa - sample_cb1)
    (forall (?sb - sample_br1) (and
      (assign (outcome_d_br1 ?sa ?sb) (+ (sample_d_br1 ?r ?sb) (outcome_d_cb1 ?sa)))
      (assign (outcome_p_br1 ?sa ?sb) (* (sample_p_br1 ?r ?sb) (outcome_p_cb1 ?sa)))
    ))
  )
)
)

```

Figure 4: CB1 from the Enhanced PDDL model

outcomes from the action. In more detail, the rationale behind the PDDL nested `forall` effects are as follows:

1. The earlier action for *CB1* defined a number of possible outcomes, the set `sample_cb1`. The duration of each of these (i.e. each `?sa - sample_cb1`) is denoted `(outcome_d_cb1 ?sa)`, and its associated likelihood is `(outcome_p_cb1 ?sa)`.
2. The action *BR1* also has a number of possible outcomes, the set `sample_br1`. The rate parameter defines the space of outcome durations and likelihoods, so for each `?sb - sample_br1` there is an outcome `(sample_d_br1 ?sb)` with likelihood `(sample_p_br1 ?sb)`.
3. The number of possible outcomes from *CB1* then *BR1* is then the cartesian product of the outcome sets, summing the durations and multiplying the likelihoods.

5 Test Case

As a proof-of-concept for the use of planning to make cost-effective investment decisions to meet a desired SLA, we shall refer to the GetLoan system (Figure 2). The scenario is as follows:

- The existing system is (easily) capable of meeting an SLA of 7 seconds 80% of the time.
- The goal is to meet an SLA of 6 seconds 90% of the time, requiring investment.
- The current mean delays for each of *CB*, *BR*, *BL* and *LE* are 0.3, 0.9, 0.3 and 1.2 seconds, respectively;
- The delay for each of these can be reduced by 20%, with respective costs of £10k, £15k, £8k, and £18k.

We proceed to make a PDDL model of this, following Section 4.2. We take 10 samples from each activity's time distribution using the baseline delays, adding these to the model with cost 0. Then, reducing the delays by 20%, we take a further set of samples and add these as alternatives, incurring the relevant cost. The plan produced is as follows, with a total investment cost of £23k:

1. CB1, baseline rate
2. BR1, improved rate
3. BL1, improved rate
4. LE, baseline rate
5. BL2, improved rate
6. BR2, improved rate
7. CB2, baseline rate

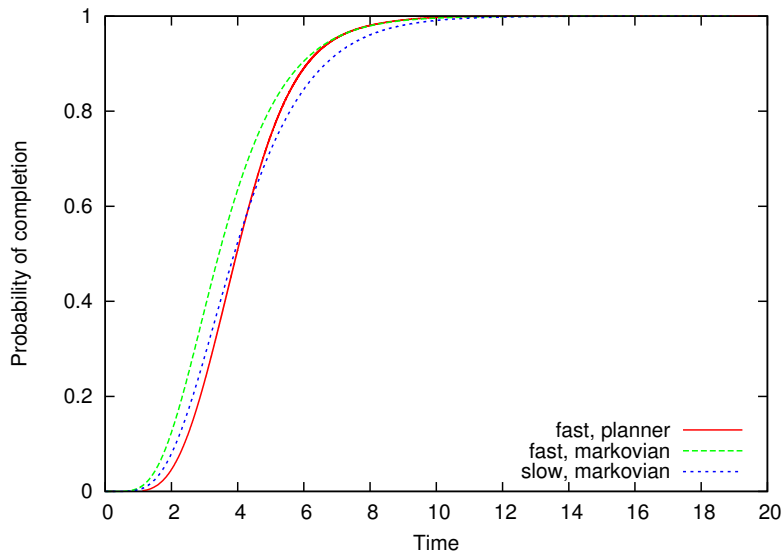


Figure 5: CDFs of the Performance of the GetLoan System

The time taken to find the plan was two seconds, and is known to be cost-optimal, with respect to the planning model: if we add an additional goal that cost is less than 23k, no solution plan can be found.

Bearing in mind that the outcome distribution used by the planner is an approximation, we can then update the original PEPA model to validate whether the investment decisions made by the planner are able to meet the new SLA. We performed a Markovian response-time analysis of the PEPA model using the `ipclib` [1] PEPA library, and the results for the original and updated models are shown in Figure 5 (the ‘slow, markovian’ and ‘fast, markovian lines’, respectively). The 6 second lies at the 90.544%th percentile of the updated CDF, indicating the investment has reached the desired SLA. Also shown on the graph is the estimated CDF produced by the planner for the outcomes of the above plan — as can be seen, it is pessimistic, but within a few percentage points of the true CDF in the upper quartile.

6 Conclusions

In this paper we have shown how PEPA modelling and Planning can be usefully combined to provide decision-support for service-oriented system configuration. By using a PEPA model to accurately reflect the uncertainty in the time taken to complete each system activity, and a planner working with an approximate sampled model, the result is a CPU-time-efficient system for suggesting and validating system investment decisions. In future work, our aim is to extend the range of system models for which our approach can be used by extending the planner to support more flexible PEPA models with disjunctive activity pathways, as well as larger models with greater numbers of possible activity configurations.

Acknowledgements

Amanda Coles is supported by the EPSRC project EP/G023360/1, “Modelling Planning Problems”. Andrew Coles is supported by SICSA, the Scottish Informatics and Computer Science Alliance. Stephen Gilmore is supported by the EU FET-IST Global Computing 2 project SENSORIA (“Software Engineering for Service-Oriented Overlay Computers” (IST-3-016004-IP-09)).

References

- [1] Allan Clark. The `ipclib` PEPA Library. In Mor Harchol-Balter, Marta Kwiatkowska, and Miklos Telek, editors, *Proceedings of the 4th International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 55–56. IEEE, September 2007.
- [2] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Temporal planning in domains with linear processes. In *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, July 2009.
- [3] M. Fox and D. Long. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *J. Art. Int. Research*, 20:61–124, 2003.
- [4] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.