COMPUTING
SCIENCE

An Empirical Study Comparing the PEPA Eclipse Plug-in and GPA Tools

Said Naser Said Kamil, Nigel Thomas
Abstract

Performance modellers rely on the accuracy and correctness of tools in order to derive results that can be trusted to give useful insights into system behaviour. However, different tools may be developed from different perspectives and apparently similar analysis may give different results in some situations. In this paper we explore the use of two tools developed to support the analysis of the stochastic process algebra, PEPA. The tools, the PEPA Eclipse Plug-in and the Grouped PEPA Analyser are shown to give consistent results in many situations, but also show some inconsistencies which may not be predictable by the modeller. Hence this paper aims to give some insight as to where the two tools differ in the context of two models of workflow based systems.
Abstract

Performance modellers rely on the accuracy and correctness of tools in order to derive results that can be trusted to give useful insights into system behaviour. However, different tools may be developed from different perspectives and apparently similar analysis may give different results in some situations. In this paper we explore the use of two tools developed to support the analysis of the stochastic process algebra, PEPA. The tools, the PEPA Eclipse Plug-in and the Grouped PEPA Analyser are shown to give consistent results in many situations, but also show some inconsistencies which may not be predictable by the modeller. Hence this paper aims to give some insight as to where the two tools differ in the context of two models of workflow based systems.

About the authors

Said Kamil is currently completing his Ph.D. in the School of Computing Science at Newcastle University. His area of investigation is the performance modelling and analysis of security protocols in Cloud Computing using stochastic process algebra. Said was awarded an M.Sc. in Computing Science with Distinction from Newcastle University in 2010.

Nigel Thomas is Deputy Head of the School of Computing Science. Nigel joined the School of Computing Science in January 2004 from the University of Durham, where he had been a lecturer since 1998. He was promoted to reader in 2009. His research interests lie in scalable performance analysis (see my personal homepage) in particular Markov modelling through queueing theory and stochastic process algebra. See also my Google Scholar Citations page. Nigel was awarded a PhD in 1997 and an MSc in 1991, both from the University of Newcastle upon Tyne. Nigel is organising the 30th UK Performance Engineering Workshop (UKPEW 2014) on 19th September 2014.

Suggested keywords

PERFORMANCE MODELLING
CLOUD COMPUTING
PROCESS ALGEBRA
PEPA
GPEPA
An Empirical Study Comparing the PEPA Eclipse Plug-in and GPA Tools

Said Naser Said Kamil and Nigel Thomas

School of Computing Science, Newcastle University, UK.
{Said.Kamil,Nigel.Thomas}@ncl.ac.uk

Abstract. Performance modellers rely on the accuracy and correctness of tools in order to derive results that can be trusted to give useful insights into system behaviour. However, different tools may be developed from different perspectives and apparently similar analysis may give different results in some situations. In this paper we explore the use of two tools developed to support the analysis of the stochastic process algebra, PEPA. The tools, the PEPA Eclipse Plug-in and the Grouped PEPA Analyser are shown to give consistent results in many situations, but also show some inconsistencies which may not be predictable by the modeller. Hence this paper aims to give some insight as to where the two tools differ in the context of two models of workflow based systems.

Keywords: Performance modelling, Cloud Computing, Process Algebra, PEPA, GPEPA.

1 Introduction

Stochastic process algebra, such as PEPA [1], has developed into a powerful approach for performance modelling since their inception in the 1990s. Process algebra offer a number advantages for modelling over other formalisms, most notably their parsimony and explicit compositional nature, which makes specifying models straightforward and facilitates the solution of models with large state spaces. One of the key aspects which has made PEPA popular is the provision of an excellent toolset, which has developed considerably over the years from a parser and simple Markov chain solver (the PEPA Workbench [4]), to a complex tool with many analysis options and debugging features (the PEPA Eclipse Plug-in [3]). More recently an alternative tool, the Grouped PEPA Analyser (GPA) [8],[9], has been developed based on a slightly modified form of PEPA, known as Grouped PEPA (GPEPA), where multiple instances of replicated components are organised into groups to facilitate analysis. GPA exploits novel results based on fluid approximations of PEPA models [5], [7]. GPA supports a number of algorithms to derive solutions based on ordinary differential equations (ODEs) and stochastic simulation, although it does not support Markov chain analysis. Similar ODE and simulation techniques are supported in the PEPA Eclipse Plug-in, albeit with less extensive analysis, and so a comparison between the two tools is clearly of interest to any active, or potential, PEPA modeller.
The main aim of this paper is to show a comparison between PEPA Eclipse plug-in and GPA tools when predicting system characteristics in two case studies, which consider simple and complex workflows. Workflow analysis has become a significant topic in performance analysis due to the use of workflows in highly scalable computing applications, such as scientific computing and cloud computing. The first case study, drawn from our own work on cloud security, presents a simple workflow with sequential services and processes. The second case study, taken from the literature, presents a more complex workflow that has multiple branches and loops.

This paper begins with brief descriptions of PEPA and GPEPA with introductions to the PEPA Eclipse Plug-in and GPA tools. This is followed by two case studies, a multi-level security model and a credit application model. Each study will demonstrate differences in model descriptions, problems and the obtained results. We end with some conclusions.

2 Tools

2.1 PEPA Eclipse Plug-in

Performance Evaluation Process Algebra (PEPA) is a high level modelling language. Hillston [1] stated that This language has been developed to investigate how the compositional features of process algebra might impact upon the practice of performance modelling. Systems are modelled in PEPA as interacting components through a set of activities [2]. The PEPA Eclipse Plug-in [3] is a tool that can be used to specify and analyse PEPA models. Several types of analysis are supported, including Continuous Time Markov Chain (CTMC), Steady State Analysis, Ordinary Differential Equation (ODE) and Stochastic Simulation Algorithms (SSA). The variety of analysis techniques provided by the PEPA Eclipse Plug-in allows system designers to find precise predictions of the system behaviours where the state space is not too large, as well as providing approximations when the state space is much too large for conventional solution.

The PEPA Eclipse Plug-in is equipped with two scalable analysis approaches, stochastic simulation (via Gillespie’s method) and fluid flow approximation by means of ordinary differential equations [5]. These techniques not only allow much larger systems to be considered, but they also facilitate transient analysis.

Analysis types A number of analysis types are provided by PEPA Eclipse plug-in tool for analysing PEPA models:

Static Analysis: Static analysis is responsible for checking the possible syntax errors in the model description. An example of this type of analysis an error message and warning that arise when a rate is declared but not used in the model. Furthermore, static analysis can detect the potential deadlocks [3].
**Markov Chain analysis:** Continuous Time Markov Chain (CTMC) analysis is used to derive the state-space and then generates all possible states of a model being analysed. An aggregation technique provided by PEPA Eclipse plug-in tool can reduce the state space of a model, to decrease the impact of state space explosion to some extent. Steady-state analysis is the next stage of CTMC, which used to calculate the steady state probability distribution via an appropriate solver chosen from the list provided. Consequently, a performance evaluation will be displayed using a number of metrics (i.e. Utilisation, Throughput and Population).

**ODE analysis:** Ordinary Differential Equation (ODE) approximation is a method for analysing the performance of systems provided by PEPA tool that can be used to conduct a scalable analysis. ODEs can be used to accurately predict the performance of large systems with many replicated components, but are generally inaccurate for smaller systems. In addition, ODE analysis implements transient analysis and steady state, unlike CTMC that provides only steady state analysis. ODE analysis is used to cope with the well-known state space explosion.

**Stochastic Simulation:** The key feature of the Stochastic Simulation Algorithm (SSA) is allowing analysing systems in realistic time, based on Gillespies approach. SSA is concerned with the number of activities within a model rather than the number of components. Simulation analysis can provide an accurate approximation, depending on setting an appropriate number of replication along with sufficient time to allow a system to reach its steady state. In general, more accuracy requires greater computation time.

### 2.2 GPEPA

Grouped PEPA (GPEPA) is a small extension of PEPA. GPEPA is aimed to generalise and simplify the methods based on Ordinary Differential Equation analysis. The limitations that related to which type of models can be subject to ODE analysis has been extended [7]. A model population is translated explicitly by GPEPA through a tool named Grouped PEPA Analyser (GPA) [8]. A formal definition of GPA tool is provided by [9], where it is defined as a tool that generates and numerically solves systems of differential equations approximating (higher order) moments of stochastic processes described in a variant of the PEPA stochastic process algebra. GPA is a command line tool that takes a model description as input along with a specification of the type of analysis to be conducted. According to Hayden [7] several advantages can be provided by GPA such as: variance, passage-time distribution via higher-moment information and scalable analysis through rapid performance calculation.

**Analysis types.** A variety of analysis techniques provided by GPA:
ODE analysis: A description of first order moment ODE analysis has been provided above. The following code shows two examples of how to specify the ODE analysis using GPA.

- Multi-level Security Model
  - The ODEs analysis (First order moment)

  ODEs (stopTime = 200.0, stepSize = 0.1, density = 10){
  E[Services:Service0], [Services:Service1], E[Services:Service2],
  E[Services:Service3]; E[Resources:Private],E[Resources:Public];}

- Credit Application Model
  - The ODEs analysis (First order moment)

  ODEs(stopTime = 2500.0, stepSize = 0.1, density = 10){
  E[Cs:C0], E[Cs:C1], E[Cs:C2], E[Cs:C3], E[Cs:C4],
  E[Cs:C5], E[Cs:C7], E[Cs:C6C8], E[Cs:C9], E[Cs:C10],
  E[Cs:C11], E[Cs:C12], E[Cs:C11b], E[Cs:C12b], E[Cs:C13],
  E[Cs:Cend], E[Resources:Resource1], E[Resources:Resource2],
  E[Resources:Resource3];}

Variance (Second order moment): GPA provides the variance analysis, which is a feature that does not exist in the PEPA Eclipse Plug-in. This type of analysis helps to gain more insight into the prediction of the obtained results, for instance the mean prediction validity can be evaluated through the variance [7]. ODE analysis is concerned about the average behaviour; the variance is how much certain actions vary around the mean. The following lines of code are used to plot the variance, which is the second order moment ODEs that generated by gpanalysers. Whereby they can be used within the ODE and the simulation analysis.

- Multi-level Security Model, the Variance (Second order moment of ODEs).

  Var[Services:Service0], Var[Services:Service1],
  Var[Services:Service2], Var[Services:Service3];

- Credit Application Model, the Variance (Second order moment of ODEs)

  Var[Cs:C0], Var[Cs:C1], Var[Cs:C2], Var[Cs:C3],
  Var[Cs:C4], Var[Cs:C5], Var[Cs:C7], Var[Cs:C6C8],
  Var[Cs:C9], Var[Cs:C10], Var[Cs:C11], Var[Cs:C12],
  Var[Cs:C11b], Var[Cs:C12b], Var[Cs:C13], Var[Cs:Cend];

Stochastic Simulation: The basic functionality of simulation in GPA is the same for the PEPA Eclipse Plug-in introduced above. In GPA the following lines of code are needed in the command line.

- Multi-level Security Model
Simulation(stopTime = 200.0, stepSize = 0.1, replications = 10000){ E[Services:Service0], E[Services:Service1], E[Services:Service2], E[Services:Service3];}

– Credit Application Model

Simulation(stopTime = 2500.0, stepSize = 0.1, replications = 10000){ E[Cs:C0], E[Cs:C1], E[Cs:C2], E[Cs:C3], E[Cs:C4], E[Cs:C5], E[Cs:C7], E[Cs:C6C8], E[Cs:C9], E[Cs:C10], E[Cs:C11], E[Cs:C12], E[Cs:C1b], E[Cs:C12b], E[Cs:C13], E[Cs:Cend], E[Resources:Resource1], E[Resources:Resource2], E[Resources:Resource3];}

Accurate Simulation: Accurate Simulation is a stochastic simulation that captures further insight in the system performance. Although the name implies that the standard simulation method is not accurate, in fact accurate simulation is so called because it is more controllable by the user. It takes more parameters, stop time, step size, confidence interval, maximum relational confidence interval width and batch size. The following commands show how GPA can execute Accurate Simulation analysis and its second order moment, to evaluate the model under assessment.

– Credit Application Model example

AccurateSimulation(stopTime = 2500.0, stepSize = 1.0, CI = 0.95, maxRelCIWidth = 0.5, batchSize = 10) { E[Cs:C0], E[Cs:C1], E[Cs:C2], E[Cs:C3], E[Cs:C4], E[Cs:C5], E[Cs:C7], E[Cs:C6C8], E[Cs:C9], E[Cs:C10], E[Cs:C11], E[Cs:C12], E[Cs:C11b], E[Cs:C12b], E[Cs:C13], E[Cs:Cend], Var[Cs:C0], Var[Cs:C1], Var[Cs:C2], Var[Cs:C3], Var[Cs:C4], Var[Cs:C5], Var[Cs:C7], Var[Cs:C6C8], Var[Cs:C9], Var[Cs:C10], Var[Cs:C11], Var[Cs:C12], Var[Cs:C11b], Var[Cs:C12b], Var[Cs:C13], Var[Cs:Cend];}

3 Case Study

3.1 Multi-level Security Model

In the multi-level security model presented by Watson [10] a set of valid deployment options was evaluated on the cost of CPU, Data Storage and Data Transfer. The model presented here depicts one such deployment option for a simple sequential workflow based on a health-care application. The aim is to find an optimal deployment over federated clouds [6]. Fig. 1 shows the workflow partitioned into two parts. Read and Anonymize services are deployed on private cloud as they are assumed to have sensitive information, whilst Analyse and Write are distributed on a public cloud.
PEPA Eclipse Plug-in

Model description: In this model there is a component which represents the workflow and components that represent the public and private cloud services. By considering Option 1 that depicted in Fig. 1 (i.e one of the valid deployment options that given by [10]) and investigating different capacities of public cloud servers, it is possible to explore which configuration offers the best overall performance. The workflow components that shown in Fig. 1, is specified as a simple sequential flow, as follows:

\[
\begin{align*}
\text{Service}_0 & \triangleq (\text{readData}, r).\text{Service}_1 \\
\text{Service}_1 & \triangleq (\text{anonymize}, s).\text{Service}_2 \\
\text{Service}_2 & \triangleq (\text{analyze}, t).\text{Service}_3 \\
\text{Service}_3 & \triangleq (\text{writeResult}, r).\text{Service}_0
\end{align*}
\]

The private and public cloud components are then specified as follows.

\[
\begin{align*}
\text{Private} & \triangleq (\text{readData}, r).\text{Private} + (\text{anonymize}, s).\text{Private} \\
\text{Public} & \triangleq (\text{analyze}, t).\text{Public} + (\text{writeResult}, r).\text{Public}
\end{align*}
\]

The system equation for the model of options 1 is given as,

\[
\text{System} \triangleq \text{Service}_0[N1] \otimes \left( \text{Private}[N2] || \text{Public}[N3] \right)
\]

Where \( N1 = 20, N2 = 1, N3 = 15 \) and the cooperation set \( L = \{ \text{readData}, \text{anonymize}, \text{analyze}, \text{writeResult} \} \). The rates used are \( r = 1, s = 0.1 \) and \( t = 0.001 \).

Problems: The PEPA Eclipse Plug-in has some limitations in deriving the steady state for large systems, because of well-known state-space explosion (i.e. by adding more components the state space grows exponentially). This results in a java heap error message, as there is no more space in the memory to accommodate the state space. This has been reported by several researches such as: [3], [5],[6],[11],[12]. However, using the ODEs analysis can scale with the large systems and can give a good approximation in reasonable processing time. Stochastic Simulation analysis can provide a more precise approximation, although it takes a longer processing time, which may be considered to be computationally costly, depending on the size and the complexity of the modelled system.
Results: Fig. 2 shows the transient evolution of the system before steady state is reached. In this instance the number of workflow instances $N_1 = 20$. The graph shows the populations of each service type, i.e. the number of workflows that are performing each action at any given time. Although, two types of analysis are shown in Fig. 2 ODE analysis (solid lines) and the simulation analysis (dotted lines), the results look fairly similar. In Fig. 2 the population of Service1 is decreasing steadily over time as Anonymize actions are completed, with the population of Service2 increasing correspondingly. Additionally, the throughput here is fairly low, so very few workflow instances are completing. In [6] comprehensive experiments have been conducted showing that, when the service capacity is doubled the situation changes due to the significant increase in throughput. This means that, the Service3 population is levelling off as more workflow instances complete. Thus the system is approaching steady state much more rapidly than in the figure below.

Fig. 2. ODEs transient analysis and simulation transient analysis of the Multi-level security Model from the PEPA Eclipse Plug-in, the solid line indicating ODEs (stop Time=200) and the dotted line representing the Simulation (stop Time=200, replications=10000).

GPA

Model description: Initially, GPA is built upon the syntax of PEPA; therefore the model description is same as the model that presented in the previous section. However, the system equation is slightly changed, where GPA uses a grouped component and curly brackets.

$$\text{System} \triangleq \text{Services}\{\text{Service}_0[N_1]\} \otimes \text{Resources}\{\text{Private}[N_2] | \text{Public}[N_3]\}$$

Problems: GPA does not calculate the direct solution of CTMC, where it relay on the ODEs analysis instead of that. This means that, it can scale up with the bigger system regardless of the number of components or the number of states derived. In Fig. 4 the variance analysis for this model generated by GPA shows a negative variance. Although, the tool allows modellers to do more investigation, this result is noticeably wrong and reduces confidence in other estimates of variance for other parameters. On the other hand, there are other situations
where the variance estimate captures performance that would not be possible using other methods.

Results: The approximation of system behaviour that generated via GPA for the first order moment shows the same performance as the PEPA Plug-in, shown in Fig. 3. Fig. 4 and Fig. 5 show the variance. One possible explanation of the negative variance shown in Fig. 4 is that when the population of a component becomes very small the discrete time step used to compute the next value is such that the linear projection from the ODEs becomes negative. All subsequent calculations are then made relative to a negative value, which is not detected in the calculation. Having both calculations of variance allows the modeller to observe differences and detect such problems visually; although it would clearly be preferable if the tool gave a warning in such cases.

![Fig. 3. ODEs transient analysis and Simulation transient analysis of the Multi-level security Model from GPA, the solid line indicating ODEs (stopTime = 200.0, stepSize = 0.1, density = 10) and the dotted line representing the Simulation (stopTime = 200.0, stepSize = 0.1, replication = 10000).](image)

![Fig. 4. The ODEs Variance of the Multi-level security Model, (stopTime = 200.0, stepSize = 0.1, density = 10).](image)

![Fig. 5. The Simulation Variance of the Multi-level security Model, (stopTime=200.0, stepSize=0.1, replication=10000).](image)
3.2 Credit Application Model

In order to consider a more complicated workflow a PEPA model has been studied based on a credit application model originally specified in YAWL\(^1\) [13]. This model has branching, loops and multiple flows between tasks. The credit application model, shown in Fig. 6, can be described as follows: after the receiveApplication task is completed, checkForCompleteness will take place. Based on the information received whether it was incomplete or complete, the task will proceed either to behave as getMoreInfo task or checkLoanAmount task. The former choice will wait until adding more information and then check again if the application is complete or not. While the latter choice means that the task checkLoanAmount is enabled, where it will proceed to perform ChecksForLargeAmount or perform ChecksForSmallAmount based on the stated loan amount. Hence, the result of the task makeDecision will direct the flow of the system whether to notifyRejection and the process is ended or startApproval which is itself has two processes. notifyAcceptance and deliverCreditCard then finally the system completing completeApproval task and the process ends.

\[C_0 \equiv (\text{receiveApplication}, r).C_1\]
\[C_1 \equiv (\text{checkForCompleteness}, p_1 \cdot r_1).C_4 + (\text{checkForCompleteness}, (1 - p_1) \cdot r_1).C_2\]
\[C_2 \equiv (\text{getMoreInfo}, r_2).C_3\]
\[C_3 \equiv (\text{checkForCompleteness}, p_2 \cdot r_3).C_4 + (\text{checkForCompleteness}, (1 - p_2) \cdot r_3).C_2\]
\[C_4 \equiv (\text{checkLoanAmount}, (1 - p_3) \cdot r_4).C_5 + (\text{checkLoanAmount}, p_3 \cdot r_4).C_7\]
\[C_5 \equiv (\text{performChecksForLargeAmount}, r_5).C_6C_8\]
\[C_7 \equiv (\text{performChecksForSmallAmount}, r_6).C_6C_8\]

\(^1\) http://www.yawlfoundation.org/
\[ C_0 \equiv (\text{makeDecision}, (1 - p_4) \cdot r_7).C_9 + (\text{makeDecision}, p_4 \cdot r_7).C_{10} \\
C_9 \equiv (\text{startApproval}, (1 - p_5) \cdot r_9).C_{11} + (\text{startApproval}, p_5 \cdot r_9).C_{12} \\
C_{11} \equiv (\text{notifyAcceptance}, r_{10}).C_{12b} \\
C_{12b} \equiv (\text{deliverCreditCard}, r_{11}).C_{13} \\
C_{12} \equiv (\text{deliverCreditCard}, r_{11}).C_{11b} \\
C_{11b} \equiv (\text{notifyAcceptance}, r_{10}).C_{13} \\
C_{13} \equiv (\text{completeApproval}, r_{12}).C_{end} \\
C_{10} \equiv (\text{notifyRejection}, r_8).C_{end} \\
C_{end} \equiv (\text{end}, r_{end}).C_{end} \]

The deployment process is modelled to be over a set of resources: Resource1 and Resource3 are assumed to be public clouds (elastic resource), while Resource2 is a private cloud (constrained resource). Another key point that needs to be clarified is that the deliverCreditCard action is an offline action, and therefore it has been removed from the deployment process as well as to the system equation. The partitioning of workflow tasks and the distribution onto federated clouds is built upon two motivations. First, allocating tasks that require more processing time and resources over public clouds to exploit high performance, availability and scalability; and secondly it has been assumed that the tasks that have high level of security will be deployed on the private cloud. The resource components are modelled as follows:

\[ \text{Resource}_1 \equiv (\text{receiveApplication}, r).\text{Resource}_1 + (\text{checkForCompleteness}, r_1).\text{Resource}_1 + (\text{getMoreInfo}, r_2).\text{Resource}_1 + (\text{checkLoanAmount}, r_4).\text{Resource}_1 + (\text{performChecksForSmallAmount}, r_6).\text{Resource}_1 + (\text{makeDecision}, r_7).\text{Resource}_1 \]
\[ \text{Resource}_2 \equiv (\text{startApproval}, r_9).\text{Resource}_2 + (\text{notifyAcceptance}, r_{10}).\text{Resource}_2 + (\text{completeApproval}, r_{12}).\text{Resource}_2 + (\text{notifyRejection}, r_8).\text{Resource}_2 \]
\[ \text{Resource}_3 \equiv (\text{performChecksForLargeAmount}, r_5).\text{Resource}_3 \]

Finally, the system equation displays the cooperation between workflow instances and resources over the set \( L \) in parallel.

\[ \text{System} \equiv C_0[N1] \parallel C_2[N2] \parallel C_3[N3] \parallel C_4[N4] \]

Where \( N1 = \{1000\}, N2 = \{35\}, N3 = \{15\}, N4 = \{50\} \) and the set \( L = \{\text{receiveApplication}, \text{checkForCompleteness}, \text{getMoreInfo}, \text{checkLoanAmount}, \text{performChecksForLargeAmount}, \text{performChecksForSmallAmount}, \text{makeDecision}, \text{startApproval}, \text{notifyAcceptance}, \text{completeApproval}, \text{notifyRejection}\} \). The following table displays the rates that have been used in the model.
Table 1. Credit Application Model rates.

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>0.9</td>
<td>r6</td>
<td>0.5</td>
<td>r_end</td>
<td>0.1</td>
</tr>
<tr>
<td>r</td>
<td>0.065</td>
<td>r7</td>
<td>0.065</td>
<td>p1</td>
<td>0.5</td>
</tr>
<tr>
<td>r1</td>
<td>0.06</td>
<td>r8</td>
<td>0.1</td>
<td>p2</td>
<td>0.8</td>
</tr>
<tr>
<td>r2</td>
<td>0.05</td>
<td>r9</td>
<td>0.025</td>
<td>p3</td>
<td>0.2</td>
</tr>
<tr>
<td>r3</td>
<td>0.1</td>
<td>r10</td>
<td>0.09</td>
<td>p4</td>
<td>0.4</td>
</tr>
<tr>
<td>r4</td>
<td>0.025</td>
<td>r11</td>
<td>0.07</td>
<td>p5</td>
<td>0.8</td>
</tr>
<tr>
<td>r5</td>
<td>0.009</td>
<td>r12</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problems: An error has occurred when implementing the simulation analysis upon the credit application model; specifically when the stop time was set to 2500 or higher. Therefore, the stop time is fixed to 2400. In this time, the tool can successfully complete the simulation analysis and gives the corresponding graph, but a warning message is shown which states that the simulation has not converged after 10000 replications.

Results: Both the approximation of ODE transient analysis and the simulation of the Credit Application Model are shown in Fig. 7. The graph shows the system evolution from $C0$ (receiveApplication) until completing all the tasks within the workflow instances by reaching $C_{end}$. The simulation and ODE results are consistent, although showing some small variations.

![Fig. 7. ODEs transient analysis and Simulation transient analysis of Credit Application Model, the solid line indicating ODEs (stopTime = 2400) and the dotted line representing the Simulation (stopTime=2400, replication=10000).](image)

GPA

Model description: The Credit Application Model description in GPEPA is identical to the model above, with the exception of the system equation, where a grouped component is used.

$$System \overset{def}{=} Cs[C_0[N]] \bowtie Resources[Resource_1[M_1] \\
| Resource_2[M_2] | Resource_3[M_3]]$$
Where \( N = \{1000\} \), \( M_1 = \{35\} \), \( M_2 = \{15\} \), \( M_3 = \{50\} \) and the cooperation set \( L = \{\text{receiveApplication, checkForCompleteness, getMoreInfo, checkLoanAmount, performChecksForLargeAmount, performChecksForSmallAmount, makeDecision, startApproval, notifyAcceptance, completeApproval, notifyRejection}\} \).

Problems: The intention was to have a separate component to depict arrivals into the workflow. Although this was possible in the PEPA Eclipse Plug-in, it could not be made to work in GPA as a grouped component. This restriction is understandable given that having a mix of single sequential components and grouped components would not lead to good approximations using ODEs. Therefore, we decided to use this slightly simpler model which works in both the Eclipse Plug-in and GPA.

Results: The first set of experiments demonstrates the evolution of the system in terms of its end-to-end behaviour, starting from \( C_0 \) (ReceiveApplication) and finishing at \( C_{\text{end}} \), when all tasks are completed. Fig. 8 shows the population in the Credit Application Model, it is obvious that the approximation of the ODE is very close to the estimates of the Simulation. Nevertheless, in both Fig. 9 and Fig. 10 there are noticeable differences in respective variance, especially in the population of \( \{C_4, C_5 \text{ and } C_9\}\).

![Fig. 8. ODEs transient analysis and Simulation transient analysis of Credit Application Model, the solid line indicating ODEs (stopTime=2500, stepSize=0.1, density=10) and the dotted line representing the Simulation(stopTime=2500, stepSize=0.1, replication=10000).](image-url)
Fig. 9. The ODEs Variance of Credit Application, (stopTime=2500, stepSize=0.1, density=10).

Fig. 10. The Simulation Variance of Credit Application, (stopTime=2500, stepSize=0.1, replication=10000).

Accurate Simulation: In the second set of experiments, the Accurate Simulation analysis is examined with the aim to explore more insight how the system will behave under certain circumstances. Table 2 shows the outcomes of a number of Accurate Simulation experiments. Five different step size values have been used with different batch sizes (10, 50, 100, and 200).

Table 2. Accurate Simulation of Credit Application Model.

<table>
<thead>
<tr>
<th>Step Size</th>
<th>Batch Size</th>
<th>Number Of Replication</th>
<th>Execute Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>25018</td>
<td>288</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>50140</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>30280</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>31560</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>30348</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>26840</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>21480</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>106160</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>41178</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>80040</td>
<td>249</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>158180</td>
<td>548</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>13160</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>9978</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>17990</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>13280</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>90960</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>10</td>
<td>5888</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1540</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>880</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>960</td>
<td>103</td>
</tr>
</tbody>
</table>
Fig. 11. Accurate Simulation of Credit Application Model (stopTime = 2500.0, stepSize = 1.0, CI = 0.95, maxRelCIWidth = 0.5, batchSize = 10).

Fig. 12. Accurate Simulation Variance of Credit Application Model (stopTime = 2500.0, stepSize = 1.0, CI = 0.95, maxRelCIWidth = 0.5, batchSize = 10).

Fig. 13. Accurate Simulation of Credit Application Model (stopTime = 2500.0, stepSize = 10.0, CI = 0.95, maxRelCIWidth = 0.5, batchSize = 100).

Fig. 14. Accurate Simulation Variance of Credit Application Model (stopTime = 2500.0, stepSize = 10.0, CI = 0.95, maxRelCIWidth = 0.5, batchSize = 100).

Fig. 11 and Fig. 13 show the accurate simulation analysis that done on the Credit application model using the parameters given in Table 2. It has been noticed that, even though both Fig. 11 and Fig. 13 have different parameters, the approximations of system performance are identical. However, a slight difference can be seen in the variance of the system shown in Fig. 12 and Fig. 14. Clearly, increasing the step size can affect the estimate of variance.

4 Conclusion

This paper has explored performance modelling in small and relatively large scale systems by means of the PEPA Eclipse Plug-in and GPA tools. This work aims to compare two different methods that used for modelling systems as well as to get more insight into system behaviour. This research is motivated by the use of workflow models from a cloud based healthcare application and a credit application model. In both cases the systems are approximated using two types of analysis specifically Ordinary Differential Equations (ODEs) besides Stochastic Simulation. In addition to this, accurate simulation provided by GPA is examined in the second case.
The results show that both tools provide consistent estimates of first order population measures. This should not be surprising, given that both tools are implementing the same underlying methods in this case. The PEPA Eclipse Plug-in also provides direct solution of the CTMC. Thus, if the state space is sufficiently small (or parameters are set to make it so) then the accuracy of the ODE approximation can be investigated. However if the state space of the model is large, then solution of the CTMC is not possible. GPA provides additional functions to derive higher order moment approximations. In this paper we have observed that the variance calculated through the ODEs can be highly inaccurate, even going so far as to give negative results. In addition, the variance calculated by the ODEs and stochastic simulation can be very different. This latter point is perhaps not surprising, as they are clearly entirely different calculations. However, to the novice user being presented with two radically different results which appear to be trying to predict the same measure does inspire confidence. GPA also provides a second, more adaptable form of simulation, known as accurate simulation. We have shown that by varying batch size and step size we can significantly influence both the execution size and estimations of variance. What is not clear in using the tool is what choice of step size or batch size is needed in order to gain the most accurate results from the accurate simulation.

In conclusion we found the PEPA Eclipse Plug-in to be relatively easy to use and intuitive. The results gained were clear and fairly consistent. GPA provides additional functionality, but controlling and understanding what the modeller should do was far from obvious. As such GPA is clearly a more specialist tool which may provide additional insight to an experienced user, but may also be confusing and reduce confidence when results are apparently contradictory.

References