18 Comparison of Techniques

18.1 Introduction

In this note we review the various approaches to representing a system which we have considered during the course, and try to identify their relative strengths and weaknesses. In doing so we gain some insight into when it is appropriate to use each of the techniques. During the course the following ways of representing the performance related aspects of the behaviour of computer and communication systems have been discussed:

- Operational laws
- Markov processes, and related high level modelling paradigms:
  - Stochastic Petri Nets, particularly GSPN
  - Queues and Queueing Networks
  - Stochastic Process Algebras, particularly PEPA
- Simulation
- Measurement

The criteria on which we will compare the different approaches are the following:

- Time and skill requirements
- Expressiveness/assumptions
- Diagrammatic representation
- Distinguishing resources
- Solution
- Deriving performance measures
- Particular strengths or weaknesses

However, you should note that not all these criteria are applicable to all the approaches, and so they will not all be considered in every case.

18.2 Operational laws

Time and skill requirements The key advantage of operational laws is their minimal requirements in terms of time and skill. This makes them especially useful for calculating bounds on performance measures rather than specific values. For example, worst and best timing assumptions can be quickly evaluated to give an estimate of the range of a performance measure for a system. The resulting bounds can contribute significantly to the analyst's understanding of the system. The calculations involved are simple, and can often even be done by hand.

Expressiveness/assumptions The assumptions which are necessary for the application of the operational laws are that the system is *job flow balanced* (*conservation of work*), and that the workload and the system are *homogeneous.*

The extremely abstract view taken of the system means that the expressiveness is poor: many aspects of system behaviour cannot be represented. For example, it is impossible to include any notion of relative ordering between jobs, synchronisation, or exclusive access. This can be viewed both as a strength and as a weakness.

Distinguishing resources The formulation of laws in terms of jobs and resources makes a clear distinction between active and passive entities within the system.

Solution and deriving performance measures The calculations involved are simple, and can often even be done by hand and unlike other techniques there is no distinction between solving the model and deriving performance measures.

Particular strengths or weaknesses The main strength of the operational laws is the speed and ease with which they can be formulated and evaluated.

18.3 Markov Processes

Time and skill requirements Modelling directly at the level of the Markov process is time consuming and error-prone for all but the simplest models, even with software support. However no sophisticated modelling techniques need to be learned and the solution procedure is a straightforward application of linear algebra, once the global balance equations have been formulated.

The time required to develop a model is often greatly reduced if a high-level modelling paradigm is used rather than working directly at the level of the Markov process.

Expressiveness/assumptions Perhaps the most important assumption for a Markov process is the *memoryless property*: that future behaviour is independent of past behaviour. In some senses this assumption does not seem unreasonable for essentially deterministic systems like computer and communication systems but it may influence our chosen level of abstraction. However this assumption implies that the inter-event times are all exponentially distributed which is rarely found to be the case when systems are monitored and measured. For solution purposes we also assume that the process is *finite*, *time homogeneous* and *irreducible.* These are reasonable assumptions from the perspective of what we are planning to model, which do not unduly restrict the expressiveness of our models.

Thus there are few restrictions on expressiveness when we model directly in terms of states and events, except the assumption that all timings are exponentially distributed.

Diagrammatic representation The state transition diagram of a small Markov process provides a clear diagrammatic representation of the system/model. Indeed, if it is correctly labelled it contains all the information necessary to solve the Markov process and can be regarded as a formal representation. However, for models with hundreds or thousands of states, constructing and viewing the state transition diagram becomes infeasible and the generator matrix is the preferred representation of the system.

Note that the state transition diagram represents the dynamic behaviour of the system, but not its structure in terms of components or subsystems.

Distinguishing resources A Markov process consists of states, and events represented by transitions. Therefore, there is no explicit representation of any entities within the system, and no distinction made between active entities and resources.

Solution Markov processes are solved *numerically* giving the steady state probability distribution. This is an *exact* solution. The main problem with this approach is one of tractability: the generator matrix, and the steady state probability vector must be manipulated and stored throughout the solution process. This can cause memory problems even with moderately sized models. This problem is often referred to as *state space explosion*.

For all but the smallest system computer-assistance is essential. However, this does not need to involve specialist software: any general package which supports the solution of linear algebra equations can be used.

Deriving performance measures State-based and rate-based measures can be calculated directly from the steady state probability distribution. This can be formalised in terms of associating rewards of appropriate values with states of interest and calculating the expected values of those rewards. Defining and identifying the states of interest can be difficult, using this direct approach, for large models. Other measures, such as response times, are usually derived using Little's law.

Particular strengths and weaknesses From a model construction point of view the main weakness of all Markov process based modelling techniques is the dependence on the exponential distribution for all inter-event times or delays. However, this is also the main strength of the approach from a model solution point of view—only memoryless systems can be solved directly in terms of the global balance equations and the memoryless property derives from the assumption of exponential delays.

State space explosion and problems of size are the major handicaps to the extensive use of Markovian based techniques in practice, and extensive research has been dedicated to finding efficient approaches to solving these models.

18.3.1 Stochastic Petri nets

Time and skill requirements The time required for model construction is generally found to be greatly reduced using a high level modelling paradigm such as stochastic Petri nets. However, some additional skill is required because the modeller must be aware of both the notation and the semantics of the new notation. The time and skill required for solution are approximately the same as when using a Markov process directly.

Expressiveness/assumptions The Petri net notation is simple with only a few primitives. This notation can be regarded as being at a low level, close to the Markov process being specified. As a result expressiveness is similar to that for Markov processes—fairly unrestricted. Stochastic Petri net models are therefore capable of representing a large class of systems.

No additional assumptions are made for stochastic Petri net models, although conditions on the underlying Markov process, such as finiteness, time homogeneity and irreducibility, can be re-expressed in terms of the net formalism.

Diagrammatic representation Stochastic Petri net formalisms are a graphical notation: the meaning of the graphical representation is formally defined and can be used to generate the underlying Markov process, via the reachability graph. In the case of GSPN the graphical notation must include annotation to distinguish any transitions which have infinite-server firing semantics.

As in the state transition diagram, the Petri net representation captures the dynamic behaviour of the system but provides little insight into the structure of the system.

Distinguishing resources All the entities within a system are represented as tokens within a stochastic Petri net model. Therefore no distinction is made between active and passive entities.

Solution - generating the underlying Markov process Generating the underlying Markov process is achieved by generating the reachability graph of the Petri net. In the case of SPNs this is quite straightforward, but in the case of GSPNs it is complicated by the presence of immediate transitions and vanishing markings. Computer-assistance is essential for this task in all but the simplest models. Once the Markov process is generated solution proceeds numerically.

Deriving performance measures Although deriving measures is handled more or less in the same way as for Markov processes, identifying the states of interest can be easier in stochastic Petri net models. Here the states are identified by their net characteristics, which relate to system characteristics.

Particular strengths or weaknesses Stochastic Petri nets are based on a formal system description technique: untimed Petri nets. Consequently these models may also be analysed to investigate the functional, or qualitative, aspects of system behaviour. One disadvantage is that it can be deceptively easy to draw a SPN or GSPN model which looks innocent, but which very rapidly generates more states than the solution tools can readily handle.

18.3.2 Stochastic Process Algebras

Time and skill requirements For the novice, stochastic process algebras such as PEPA, are perhaps less intuitive to use than stochastic Petri nets. However, the component-based approach greatly simplifies the task of model construction by allowing the modeller to focus on each subsystem individually and then model their interactions. Moreover, the use of previously defined components can speed up model construction. Basic performance measures may be derived from the solution of the underlying Markov process without detailed knowledge of the algebra, and the time and skill requirements are approximately the same as when using a Markov process directly. But, as with stochastic Petri nets, to take full advantage of the formalism the modeller should be aware of the stochastic process algebra both as a system description technique and in terms of its underlying semantics, equivalence relations, etc. This requires quite a substantial investment of time initially.

Expressiveness/assumptions Stochastic process algebra languages are very simple: systems are described as components who undertake actions and a small number of combinators define how such components and interactions between them can be built up. Since the notation is again low level the expressiveness is close to that of Markov processes themselves. It has been formally proved that the expressiveness of PEPA is equivalent to the expressiveness of bounded SPN models. The necessary assumptions are just those for Markov processes.

Distinguishing resources Distinction between *active* and *passive* is made on an action, rather than a component, basis. This means that an individual component may be both active and passive during its lifecycle with respect to different action types. Nevertheless this gives a clear indication within the model of where the control of each action lies.

Solution - generating the underlying Markov process Generation of the underlying Markov process of any stochastic process algebra model is formally defined based on the operational semantics of the language. Tool support exists to do this automatically for most of the stochastic process algebra languages. Each state in the *derivation graph* is associated with a node of the state transition diagram of the underlying Markov process. Once the Markov process is generated solution proceeds numerically.

Deriving performance measures As with stochastic Petri nets, it is often possible to derive performance measures via reward structures, using characteristics of the model to identify those states to which rewards should be attached. This is much less error prone than examining the state space directly.

Particular strengths or weaknesses Stochastic process algebra models bring several new features to performance modelling, because of their basis in the formal description technique of process algebra. Primary amongst these is the compositional structure readily apparent within models. Not only does this aid model construction, but it can, in some circumstances, be exploited during model solution. Other novel features are equivalence relations which allow models to be compared and abstraction mechanisms which allow unnecessary detail to be disregarded. The main weakness of stochastic process algebras comes down to the problem of state space explosion. As we have seen in the course approaches to overcoming this problem are now well-developed and implemented in the software support. Essentially there are two approaches based on stochastic simulation (preferable when you need to keep track of individual entities in the model) and fluid approximation (preferable when there are large numbers of instances of the same type of components, but only average behaviour is captured).

18.3.3 Queues and queueing networks

As earlier in the course, here we only consider queueing networks which have a *product form* solution. Other queueing networks require more sophisticated techniques, relying on the skill and experience of the modeller, or must be expanded out into the related Markov process.

Time and skill requirements For many applications, queueing networks achieve a favourable balance between accuracy and efficiency. Both single queue and simple queueing network models can be constructed, parameterised and evaluated relatively easily and without detailed knowledge of the underlying theory. Since solutions for most commonly occurring Markovian queues are widely published, single queues and simple product form queueing networks can be used almost as readily as the operational laws.

Expressiveness/assumptions Queueing networks are a compact notation in which many systems may be represented concisely. The behaviour of each service centre can be simply expressed, with a few descriptors, based on the six basic characteristics used in Kendall's notation.

The penalty for the compact notation is the limited expressiveness of the language. Most notably, queueing models cannot represent systems in which more than one resource must be simultaneously retained or in which there is internal concurrency. Work has been done to extend product form queueing networks to such systems, but the results are not generally applicable.

Diagrammatic representation The diagrammatic representation of a queueing network is largely schematic, and does not have a formal interpretation in the same way that a stochastic Petri net does. However, the structure of a queueing network will often bear a close resemblance to the physical structure of the system being modelled. The diagram usually also captures a high-level view of the dynamic behaviour of the system.

Distinguishing resources The resources, within the system are clearly identified as the *servers* whereas the more passive entities are represented as customers which circulate within the network. In fact in this case the distinction between active and passive is not so clear cut, since the servers do all the work but passively wait for customers to present themselves, whereas customers actively move between resources but generally have no control over the service when they get there.

Solution and derivation of the performance measures Queueing networks are solved analytically, but generally without recourse to direct solution of the global balance equations. For most commonly occurring queues, expressions or algorithms for calculating performance measures directly from the parameters of the queue have been widely published. Product form queueing networks are such that each queue may be solved separately once the appropriate arrival rate has been found. Arrival rates at all the service centres in the network are calculated using the *traffic equations*. These have the advantage of being *linear* in the number of service centres in the network, whereas the global balance equations are *exponential* in the number of service centres in the network. This computationally efficient method of solution is largely responsible for the popularity of queueing networks for performance modelling.

Particular strengths and weaknesses Restriction to the subset of product form queueing networks implies certain assumptions about the system under study. On the one hand, these assumptions seldom are satisfied strictly. On the other hand, the inaccuracies resulting from violations of these assumptions typically are, at worst, comparable to those arising from other sources (e.g. inadequate measurement data).

18.4 Simulation

Time and skill requirements Simulation models generally are expensive to define, because this involves writing and debugging a complex computer program. They can be expensive to parameterise, because a highly detailed model typically requires a large number of parameters. Finally, they are expensive to evaluate because running a simulation requires substantial computational resources, especially if narrow confidence intervals are desired. Developing a good simulation model will often require detailed knowledge of the system being represented as well as a thorough understanding of statistical techniques.

Expressiveness/assumptions There are few restrictions on the behaviour that can be simulated, so a computer system can be represented at an arbitrary level of detail. Similarly any distribution can be used to represent delays and other distributions within the system. However, this expressiveness does not come without a cost: complex models take a long time to solve. Even a detailed representation may embody subtle assumptions about the behaviour of the system but often these assumptions will be implicit.

Distinguishing resources In the process-based view of simulation a distinction is generally made between active and passive entities within a system. This is not the case in the event-based view where all entities are only represented implicitly.

Solution A simulation is an *algorithmic abstraction* of the system, rather than an *algebraic abstraction*. This means that the model is executed to reproduce the behaviour of the system, rather than being solved to analyse the behaviour. There are several disadvantages to this: in order to ensure that a run is representative of all aspects of system behaviour a long execution may be necessary; all “results” are merely observations from this particular run; repeated executions are necessary in order to generate good estimates.

In contrast, the advantages of this approach are the greater freedom offered to model important aspects of system behaviour in detail and the ability to consider models who state space exceeds the capabilities of numerical analysis.

Deriving performance measures Because simulation models are *run* rather than *solved*, performance measures are *observed* rather than *derived*. As with all other aspects of modelling, simulation offers the modeller a great deal of flexibility in how measures can be defined. However, a single observation should not be regarded as conclusive: variance reduction techniques should be employed to generate independent observations of the measures.

Particular strengths and weaknesses The principal strength of simulation is its flexibility; its principal weakness is its relative expense. Generating transient measures is straightforward for simulation models whereas it is more difficult for Markovian models.

18.5 Measurements

In this course we have only really considered measurements of a system as a means to develop a workload characterisation and thus values for input parameters. However, it is sometimes useful to remember that a set of measurements may also be used as a representation of a system.

Time and skill requirements Measurements generally take longer than Markovian models to develop but less time than simulation models. If specific monitoring software or hardware needs to be developed, obtaining measurements becomes very demanding both in terms of skills and time. Analysing and summarising the collected data also requires knowledge of statistical techniques and, often, a substantial investment of time.

Expressiveness/assumptions Which aspects of system behaviour can be represented by measurements will be influenced by many factors, most obviously whether the system actually exists or not. Even if the system exists monitoring a real workload may be too disruptive leading to the use of synthetic workloads in isolation. Unless monitoring software is especially written, the data collected may not represent the desired characteristics of system behaviour.

Solution Although it may seem that measurements should be able to produce very accurate results this is not always true. The results may be inaccurate because many of the environmental parameters, such as system configuration, type of workload, and the time of the measurement, may be unique to this experiment. Moreover, if a *live* system is being measured, as in one in which users are currently active, the users may modify their behaviour because of the overhead introduced by monitoring making the system behaviour observed atypical.

Deriving performance measures Choosing performance measures to summarise measurement data can be as difficult as parameterising a performance model. Workload analysis can help to identify which parameters are significant. However, care is needed, especially when comparing two systems on the basis of measures. For example, the performance of CPUs may be compared on the basis of their throughput, measured in MIPS (millions of instructions per second). However, if one CPU has a RISC architecture and the other has a CISC architecture the comparison in terms of MIPS is meaningless.

Particular strengths and weaknesses Obviously measurements are only possible if the system, or a similar system, exists and appropriate monitoring software or hardware is available.

Using real user workloads to generate measures raises issues about the analyst's ability to control or repeat the measures; on the other hand, using synthetic workloads such as *benchmarks* raises issues of how realistic the measurements are.

All measurements are as susceptible to experimental errors, bugs or misinterpretation as the other modelling techniques.

18.6 Some concluding remarks

Skill in introducing and assessing assumptions is one of the keys to conducting a successful modelling study. In general, it is important to be explicit about the assumptions that are made, the motivations for their introduction, and the arguments for their plausibility. This allows the analyst's reasoning to be examined, and facilitates evaluating the sensitivity of the results to the assumptions.

Sometimes it is unavoidable that questionable assumptions must be introduced. For example, if inadequate measurement data is available the modeller may have to estimate parameter value on the basis of very little information. In this case, *sensitivity analysis* can be used to determine the extent to which such assumptions cast doubt on the conclusions of the study. This will commonly take the form of evaluating the model a number of times for variations in the assumption, and comparing the results. Thus the analyst is able to judge how influential the assumption is on the results of the model.

It is important for any modelling study that the modeller has a thorough understanding of the system under consideration, but also a clear understanding of the objectives of the study. The modeller's understanding of both is likely to improve as the model develops, which is one of the reasons why conducting a modelling study is often an iterative process. As a general rule of thumb, detail should only be added into a model when it is found to be necessary. Many system characteristics that would need to be represented in a fully general model may be irrelevant in a particular study. Identifying these characteristics leads to a simpler model and a simpler modelling study.

Finally, however detailed the model, and however long the modelling study has been, performance analysts should always bear in mind the limitations of their approach. In general, the results of performance analysis will be *evidence* rather than *fact*, and should be presented as such (cf. the Hubble Space Telescope discussed in Note 17).

Jane Hillston ⟨Jane.Hillston@ed.ac.uk⟩. March 15, 2017.