Predictive Modelling for Security Operations Economics♦

Mike Yearworth, Brian Monahan, David Pym
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2006-125
September 20, 2006*

Information security operations - necessary to protect the confidentiality, integrity, and availability of an organization's information systems against attacks - represent substantial investments in technologies, tools, and human resources. Typically, the relationship between the supplier of an information system and its users is regulated by a Service Level Agreement, and the supplier must determine the appropriate level of investment in operational resources in order to meet its contractual obligations whilst maintaining its economic viability. We contend that investment decisions should be based on analytic models of the behaviour of information systems in the context of the environmental threats they face. We describe a mathematical framework, together with a modelling philosophy, for capturing the structural and dynamical properties of systems and their associated security operations. We describe how a modelling tool (Demos2k) can be used to capture much of our conceptual framework, giving a detailed, experimental example. We show that our models are able to predict the economic consequences of investment decisions for security operations.

Predictive Modelling for Security Operations Economics[†]

Mike Yearworth, Brian Monahan, David Pym
Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK.

{mike.yearworth, brian.monahan, david.pym}@hp.com

11th September 2006

Abstract

Information security operations — necessary to protect the confidentiality, integrity, and availability of an organization's information systems against attacks — represent substantial investments in technologies, tools, and human resources. Typically, the relationship between the supplier of an information system and its users is regulated by a Service Level Agreement, and the supplier must determine the appropriate level of investment in operational resources in order to meet its contractual obligations whilst maintaining its economic viability. We contend that investment decisions should be based on analytic models of the behaviour of information systems in the context of the environmental threats they face. We describe a mathematical framework, together with a modelling philosophy, for capturing the structural and dynamical properties of systems and their associated security operations. We describe how a modelling tool (Demos2k) can be used to capture much of our conceptual framework, giving a detailed, experimental example. We show that our models are able to predict the economic consequences of investment decisions for security operations.

1. Introduction

1.1. The economics of security operations

Information security operations — necessary to protect the confidentiality, integrity, and availability of an organization's information systems against attacks — represent substantial investments in technologies, tools, and human resources. The objective of these investments is to allow the organization to maintain its operational (e.g., trading) activities and so security operations investment decisions must be consistent with the organization's overall operational and financial models. Indeed, estimating the return on ICT investments must be based on a desire to avoid losses arising from externalities, such as indiscriminate worms and viruses or highly focussed attacks. Such an analysis is challenging (Anderson 2001), and requires an approach to monetizing the loss of availability, integrity, and confidentiality, and to estimating the risk of occurrence. Therefore, we should no longer regard ICT security as being purely a technology issue, at least within the sphere of corporate business. Rather, it is helpful to regard ICT security as a process — something that is done to create a smooth environment for the organization's operations.

In order to understand the value of operational processes, modern corporate management makes essential use of metrics to assess impact upon business performance, so enabling an understanding of the consequences of operational decisions for key indicators, such as shareholder value. For security, the question that then arises is that of how to measure the level of *security performance* that these security-related processes achieve (Gordon & Loeb 2006). In order to formulate this question precisely, it is necessary to understand the goals of security operations and to have mechanisms for predicting the

---

outcomes of specific processes. Then it becomes possible to integrate ICT security investment decisions within an organization's overall operational and financial processes.

1.2. The use of Service Level Agreements

Typically, the relationship between the supplier of an information system and its users is regulated by a Service Level Agreement (SLA) between the internal functional units of an organization and, more usually, with external services providers. SLAs provide a contractual statement of what is to be delivered to the organization by the provider, internal or external. Contractual statements such as these thus formalize, to some extent at least, the business process relationships and their attendant expectations.

Although use of SLAs is widespread, we make an important qualification: we are concerned *only* with SLAs that can be considered to be *meaningful*: that is, those that adequately map the financial value of the performance of business processes, running on a given infrastructure, to the cost of provisioning. With respect to security, these contractual statements have commonly taken the form of pure *policy compliance* statements; for example, conformance to ISO17799, and such like. Satisfaction of requirements such as these is then audited on a regular basis, the results of which may contribute to a corporate annual report, and thus have impact upon business confidence.

1.3. A model-based approach

The approach we have taken towards understanding the relationship between information systems properties and their SLAs is based on an analytical model based approach. In particular, we focus on whether SLAs relating to security operations are 'meaningful'. This requires that we must be able to:

1. **Measure** the behaviour of the security operations system in such a way that it enables effective control. For example, one could measure the number of repairs currently underway involving operations staff, or the number of business alignment tasks in hand at this time. Such information about the current situation would typically be fed into the management decision process and thus influence future levels of activity and response;

2. **Predict** behaviour based on the use of a model of the security operations system; that is, answering hypothetical questions such as what is the impact on overall availability if we reduce or increase security operations staff by some percentage, for example.

Given our emphasis upon measurement and prediction, we also find it useful to take a control-systems perspective when constructing our models. In such a model, we will typically need to represent various business entities in terms of processes that respond to events occurring at particular rates and with particular probability. In developing our model, we have adopted a number of guiding principles derived from Open Analytics practice (HP 2006).

It is important for these models to be both tractable and intelligible. Accordingly, we need to work at a suitable level of abstraction so that it remains conceptually clear how to interpret synthetic data obtained from simulation. We have deliberately taken an approach that is necessarily phenomenological, with the advantage that is not necessary to derive an *exact* model of the system. Primarily, our model is not *too detailed*, so as to be overly complex and slow to develop, execute and maintain; equally, it is not *too simple*, either as to miss important dynamical behaviour. Also important is the *time value* of modelling — that is, for any model to be useful it must deliver results quickly enough to contribute to a financial decision making process. Clearly, the model must also carefully distinguish between cause and effect and thus separate out:

- **External elements** corresponding to the 'IT threat environment' which includes the rate of discovery of vulnerabilities, speed to develop exploits, speed to develop patches and signatures; and

2

- **Internal elements**, such as specific tasks undertaken in security operations and the speed with which these tasks are undertaken, propagation models of attacks and attack effectiveness.

1.4. The Demos 2000 language

This paper presents the development of a preliminary model of security operations using Demos2000 (henceforth Demos2k), as developed by Christodolou, Taylor, and Tofts (2000), for (indicative) prediction, based upon phenomenological parameters. Sources of data that contributed to our understanding of how to parametrize the model included costs of security operations and technologies, security metrics and threat dashboard, financial tools (roughly in line with the approach of Gordon and Loeb (2005)) and SLAs.

Demos2k is a semantically justified discrete event simulation language implemented in Standard ML and available for download under an experimental licence from HP. Appendix A contains some further discussion of Demos2k and the basic modelling concepts it support directly. Such concepts find specific uses within our model as follows:

- *entities* correspond to internal security operations tasks (test patches, clean machines, routine patching) and to external activities (exploit development, patch development);

- *resources* equate directly to the *Full Time Equivalent[1]* (FTE) security operations staff required to perform security activities;

- *constants*, drawn from distributions, often equate to metrics; that is, meaningful measurements of the system being modelled; either existing metrics (e.g., Intrusion-Detection System (IDS) detects, publicly announced vulnerabilities, IT Security Incident-Response Team cases), or others as they become feasible to measure. Rates associated with external elements will have to be best guesses based on intelligence gathering and/or threat assessments and may be supplied by external organizations;

- *variables* are used to count basic events and so forth. In the model, these values contribute to SLAs or components thereof.

The rôles of Demos2k entities, resources, constants, and variables that are described here derive from an understanding of how to model the static and dynamic components of a (security) system, as sketched in (Monahan and Pym 2006). In that work, a theoretical framework is proposed in which systems that deliver services are described by structured collections of locations, at which reside structured collections of resources, which together support the execution of processes. The properties of such a system are then described by a system of logic that exploits the structure of the locations and resources in order to account for concepts such as sharing and privacy. We return to these issues briefly in §2.1.

1.5. The prediction of performance against Service Level Agreements

The potential security SLAs developed through this work should provide a focus for the cost–benefit analysis of security operations. As a consequence we should be in a position to answer questions such as: "What would happen to performance against the security SLAs if the number of security operations FTE were reduced 10%?"; or "What would happen to the performance against security SLAs if the rate of discovery of security vulnerabilities increased by 50%?". These are either very broad or quite narrow questions depending on the scope of the SLAs, where scope can be defined by whatever criteria would

---

[1] This neatly avoids having to keep track of all the factors that translate headcount into the staff available to carry out tasks. For a specific organization, an operations manager should be able to translate back from an FTE requirement, predicted by modelling, to the necessary headcount. Some of these factors include contracted hours, shift patterns, breaks, holidays, and illness rates.

define a meaningful SLA. For example, the security SLA may refer to the availability of a critical business application running on a particular set of servers.

1.6. The focus on availability modelling

As stated, the classic objectives of security are protecting the confidentiality, integrity and availability of information assets and systems within an organization. Of these, our focus in this paper is on availability modelling. We therefore monetize lack of availability information system as a cost to the organization arising from the negative impact on the performance of business processes, and ultimately on revenue. However, it must be stressed that loss of integrity and breaches of confidentiality also lead to financial losses but these are not discussed in detail in this paper and modelling them are the subject of further research.

2. Developing a Predictive Model (and Modelling Framework)

2.1. The conceptual and mathematical framework

In this section, we explain informally the theoretical basis of our modelling technology by describing our semantic analysis of the structure and dynamics of the systems that we aim to model. We begin with a brief account of our conceptual analysis, then give a brief description of its mathematical realization in the *synchronous calculus of resources and processes* (SCRP, for short) and an associated modal logic (MBI) (Pym and Tofts 2006, Monahan and Pym 2006), and conclude with a brief summary of the extent to which our framework is partially realized by the Demos2k tool.

Our *conceptual* framework for modelling the integrity and performance of systems is based on an analysis of four key concepts.

- *Resource*. Informally, we consider resources to be the consumable, static components of a system. Examples include computer memory, processor cycles, docking bays in a port, and money, among many other similarly natural examples. Considering these examples, it is natural to require that it should be possible to *combine* resources. For one example, two pots of money may be combined to form another (larger) pot of money. For another, two regions of computer memory maybe combined provided they do not overlap. Conditions such as this latter one are known as *separation conditions* and have been studied in some detail in, for example, (Pym 2002, Ishtiaq and O'Hearn 2001, Reynolds 2002). It is also natural to be able to *compare* resources. For example, we might compare pots of money, as above, or the size of a region of computer memory. For another example, resources might be regions of a file system ordered by an access control régime such as the Bell-La Padula protection model.

- *Process*. We consider processes to be the dynamic parts of systems, which manipulate — for example, consume, move, combine — resources. For instance, the basic functions of an operating systems, or the *movements* of a boat around a dock as it arrives, is loaded or unloaded, and departs. Fortunately, there is a well-developed mathematical theory of processes (Milner 1983, Milner 1989, Pym and Tofts 2006) which is well-suited to our purposes. It is sketched very briefly below.

- *Location*. The idea of location is an intuition derived from the physical concept of *place*. For one example, regions of memory to which an operating system must write may be located on different disk drives. For another, a port may have several docking bays and the choice of which to use for any given boat may depend on the widths and depths of the access channels to the different bays. Considering the various examples of location, it seems that a model should capture the following features (Monahan and Pym 2006): there should be a notion of *sub-location*, a notion of *connection* between locations, and a notion of *zooming in and out* to consider more or less detail

of the 'map'. For more technical reasons, we also need a notion of the *product* of locations. In the security domain, a concept of location arises naturally and essentially. For example, an infection starts at some place(s) then spreads around the system following an epidemiological pattern that depends, among other things, on the connectivity of the system components. Understanding the topological properties of the system, such as connectivity, is therefore essential to determining the appropriate deployment of countermeasures.

- *Environment*. Systems, be they IT, social, business, or physical, are intended to perform functions within an environment of events. In our framework, we capture the environment of events *stochastically*. That is, we assume that given classes of events occur — that is, are incident upon the system of interest — with given probability distributions. For example, we might assume that boats arrive at a port according to a negative exponential distribution with parameter λ. Given such an assumption, it is then natural to understand the flow of resources between locations, described as a collection of processes, as a system of queues.

Our *mathematical* framework reflects the conceptual structure outlined above. Each of the key concepts, resources, process, and location, is captured mathematically by considering the basic mathematical properties that we expect of it. So, our methods are those of classical applied mathematics but using mathematical tools drawn from logic, theoretical computer science, and probability theory.

- *Resource*. We have seen that the key aspects of resource that we want to capture are unit, composition, and comparison. Mathematically, we capture these properties by requiring that resources carry the structure of a *pre-ordered, commutative, partial monoid* (subject to some mild algebraic conditions). We write

$$R \ = \ (\mathbf{R} \ , \ \circ \ , \ e \ , \ \sqsubseteq)$$

to denote the evident quadruple, consisting of an underlying set of resources ($r, \ s \ \in \ \mathbf{R}$), a composition operation ($r \circ s$), a unit (e, the identity element for composition), and a comparison relation ($r \sqsubseteq \ s$) defining the monoid. An example, corresponding to money, is given by the natural numbers combined using addition, with unit zero, and ordered by less than or equals.

For SCRP, we take sets of resources over such an underlying monoid, lifting the monoidal structure in a straightforward way.

- ○ *Composition (and unit)*. The composition of a set of resources is given by the set of compositions of elements of the underlying set. The unit is then $\{e\}$.

- ○ *Comparison*. There are some choices, one example being $R \subseteq S$ if, for all $r \in R$, there is an $s \in S$, such that $r \sqsubseteq s$.

- *Process*. Our model of process is based on a development, SCRP, of Milner's SCCS (Milner 1983) which integrates our model of resource. The basic components of SCRP can be summarized as follows:

- ○ *Actions*, which carry the structure of a monoid — so any two actions a and b can be combined to form a composite action, $a \# b$;

- ○ *Combinators*, which allow processes to be built out of actions. The combinators we take are *action prefix*, which allows the sequencing of actions, *non-deterministic choice* between processes, *concurrent composition* of processes, a *local binding* of resources to processes, and *recursion* (via constant definitions). Formally, the grammar of processes, E, is given as follows, following the order of introduction above:

$$E \ ::= \ a \ \big| \ a : E \ \big| \ E + F \ \big| \ E \times F \ \big| \ (\nu R) \, E \ \big| \ C := E \, .$$

The connection between actions and processes is expressed using a *modification function*, μ, a partial function that maps an action together with a resource to a resource.

The meaning of this syntax is given by an *operational semantics*, expressed as a collection of *inference rules*. For example, the rule for action prefix

$$\frac{\mu(a,R)\downarrow}{R,a:E \xrightarrow{\ a\ } \mu(a,R),E},$$

says that, provided the effect of the action a on the resource R be defined, then, with resources R the process $a : E$ can evolve by the action a to become the process E with resources $\mu(a,R)$ as specified by μ. Another example is given by the rule for concurrent composition,

$$\frac{R,E \xrightarrow{\ a\ } \mu(a,R),E' \quad S,F \xrightarrow{\ b\ } \mu(b,S),F'}{R \circ S, E \times F \xrightarrow{\ a\#b\ } \mu(a\#b, R \circ S), E'\!\times\!F'},$$

provided the modification $\mu(a\#b, R \circ S)$ is defined. This rule expresses how we form a concurrent process from its components. Notice that the composite resource is determined by the composition in the resource monoid, where our separation conditions can be imposed. The rules for the other combinators are expressed similarly.

Finally, we remark that *equality* between SCRP processes is given by bisimulation (Milner 1983, Milner 1989) relative to a given resource and modification function.

- *Location*. Our model of location, capturing the conceptual requirements described above, is very simple. We require a mathematical structure that supports the following:

    o A collection of locations, *L*, *L'*, *M*, *M'*, etc;

    o *Substitution* of locations, *M*[*L'/L*], of a location *L'* for a sub-location *L* of *M* . This idea requires a suitable notion of arity of location, so that substitution suitably preserves connectivity;

    o A notion of *connection* between locations *L* and *M*;

    o Finally, a product of locations.

    Directed graphs provide a simple and natural example of a model of location.

- *Environment*. As we have seen, we treat the behaviour of the environment stochastically. The simplest way to understand the relationship the stochastic environment and the process-theoretic structure of our models is via actions. An example will make things clear. Consider again the example of boats in port. We have a process that describes the movements of a boat around the port. But how do we initiate such a process? When a boat arrives at the port, according to the probability distribution for such arrivals, such as negative exponential or Poisson, the first action in a boat process is initiated. This happens for each incidence of a boat, and multiple boats execute in the system as concurrent processes (see above). Note that this gives us an example of a separation condition; concurrent boats may not share the same docking bay; their respective docking-bay resources must be separate.

The modelling language used in our examples, Demos2k, provides a *partial* realization and implementation of our mathematical model of our conceptual framework. Summarized below is the extent to which Demos2k (see Appendix A) captures our framework, concentrating on just the key points.

- *Resource*. Demos2k has several notions of resource, primarily *bins* and *resources*. Bins provide a basic form of resource that is essentially counters: processes put elements in and may (then) take

them out. They have no in-built notion of composition or comparison beyond numerical ordering. Demos2k's resources provide stocks of elements for which concurrent processes (called entities) may compete. Just as for bins, resources also have no in-built notion of composition or comparison.

- *Process.* The basic notion of process in Demos2k is the *entity*. Entities may be thought of as (recursively defined) sequences of actions. Entities manipulate resources.

- *Location.* There is no inherent notion of location in Demos2k. Location must be represented implicitly in models. In particular, the model discussed in 3.1 represents different locations using distinct names. In more complex models, the complexity introduced in this way would become difficult to manage.

- *Environment.* Demos2k supports a very wide range of probability distributions for representing the behaviour of the environment and, indeed, stochastic aspects of the internal construction and operation of models.

A process-theoretic semantics for Demos2k has been provided in (Birtwistle and Tofts 1993, 1995). Similar analyses, exploiting SCRP's built-in resource semantics, can be given in our setting.

Our mathematical framework admits also the possibility, not exploited in this paper, of providing a system of logic, tailored to our model of resources and processes, in which system properties can be expressed. The basic idea is to set up a logical judgement relating resources, processes, and their logical properties, as expressed as

$$R \ , \ E \models \phi$$

and read as the property ϕ holds — that is, is true of — of process E relative to the set R of resources. The relationship of truth between propositions, resources, and processes is defined inductively on the structure of propositions, with a case of the induction for each of the logical connectives that is supported by the semantics. A full discussion of the basics of this logic, called MBI, is provided in (Pym and Tofts 06) and is beyond our present scope. For now, however, we remark that logical assertions ϕ might be used, for example, to express access control policies for the system (model) described by the resource-process R,E, so that the judgement $R \ , \ E \models \phi$ will hold just in case the system (model) is compliant with the policy.

MBI is based on the logic of bunched implications, BI, introduced in (O'Hearn and Pym 1999, Pym 1999, Pym 2002, Galmiche, Méry, and Pym 2005), and Hennessy-Milner logic (Stirling 2001). The main point about BI is that it permits both 'additive' and 'multiplicative' logical connectives at the same level of abstraction as one another, thereby admitting, when combined with modalities in the sense of Hennessy-Milner logic, a logical analysis of the global and local properties of systems. This point may be seen quite quickly by considering the difference between the additive and multiplicative conjunctions, $\wedge$ and $*$, respectively. We have

$$R \ , \ E \models \phi \wedge \psi \ \text{ iff } \ R \ , \ E \models \phi \text{ and } R \ , \ E \models \psi,$$

which refers to the resource and process only locally. On the other hand, we have

$$R \ , \ E \models \phi * \psi \ \text{ iff } \ S \ , \ F \models \phi \text{ and } T \ , \ G \models \psi,$$

where R is the composition of S and T, and where E is bisimilar to $F \times G$. Notice here that the definition of the truth condition for $\phi * \psi$ refers to the global structure of the model of the system and, as a consequence, provides a logical characterization of concurrent composition (Pym and Tofts, 2006).

As in Hennessy-Milner logic, the connection between the logic and the dynamics of the processes is facilitated by the modalities, [a] and $\langle a \rangle$. In MBI, these modalities have both additive and multiplicative

forms, analogous to the conjunctions described above. The additives may be seen as 'temporal' and the multiplicatives as 'spatial' or 'resourceful'. These logical ideas can be further enriched with our notion of location, leading to a logical judgement of the form

$$L, R, E \models \phi .$$

which, when given with an account of how resources are located using a modification function that is parametrized on locations, is read as property ϕ is true of the process E relative to resources R located at L. The operational counterpart, depending on the same parametrized modification function, then takes the form

$$L, R, E \xrightarrow{a} L', R', E' ,$$

where $\mu(a, L, R) = (L', R')$, with suitably parametrized rules for each of the combinators.

The remaining active strand of research in this area concerns the key security concept of access control. We can consider a user or a group of users, or rôle, that is to say, a *principal*, to be a process that is being executed on a system; that is, relative to some resources that are situated at locations. Building on some ideas introduced by Abadi et al (1993) we are able to make mathematically precise their conception that impersonation — hence groups, rôles, etc — is a form of concurrent composition of the impersonated and impersonating principals. Using the ideas from logic to which we have alluded above, we can integrate specifications of access control régimes.

2.2. The modelling process

The framework in which the availability model has been developed is shown in Figure 1. We refer to the mathematical basis, as explained in §2.1, as 'SCRP/MBI + location' for brevity. Of course, this account must be embedded in a stochastic treatment of environmental and operational events, also as discussed in §2.1.
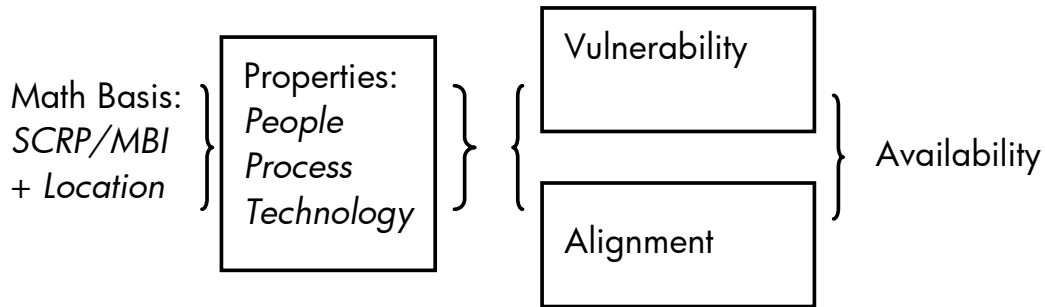
Figure 1. Framework used to derive availability model.

Demos2k has been used to develop the model described in this paper. As we have explained in 2.1, Demos2k embodies the stochastic treatment of environmental and operational events but has just an 'atomic' notion of resource, lacking the structural content of resources that allows resource composition and comparison, and also lacking a notion of location. The integration of these concepts into a basis for Demos2k-like tool is the purpose of further theoretical and implementation research on SCRP/MBI-based tools.

We have focussed initially on threat modelling and internal mitigation processes associated with vulnerabilities such as patching and fixing compromised machines. Misalignment modelling is relatively

less well-understood and ripe for further research effort. Internal alignment processes include the following: configuring new users, new devices, new customers, and new partners; updating configurations based on change requests; and non-standard business-critical fixes. Instead of modelling many individual processes, however, we have for now abstracted away into a single 'align' process that consumes operational resources for an average job-time, set by the change management request rate. The average job-time and change-management rate are two operations metrics that it would be useful to capture within the organization.

This abstraction to highly aggregated components of the system has advantages, such as conciseness, and efficiency. Moreover, such a modelling style is strongly suggested by the form of Demos2k with its lack, as discussed above, of substantial structural components (of course, we could code them in for any given model, but there would be significant loss of clarity, efficiency, and modelling rigour).

3. The Demos2k Availability Model

The focus in this paper is on modelling availability with a view to predicting performance against a meaningful SLA. Sources of downtime, or lack of availability, of an ICT system can be characterized as follows:

1. **Fix time:** time taken to repair specific components of the system that have been compromised or damaged by attack. Typically this would be cleaning or rebuilding a computer including the time taken to recover data. This source of downtime arises from *vulnerabilities*;

2. **Misconfiguration:** lack of access to a significant business[2] component of the system (e.g., a business application like an ERP system). By misconfiguration we mean any configuration of the system incompatible with operation of business processes (e.g., authentication failure, incorrect assignment of rôle) which would require security operations resources to resolve. Also note the principle of constant change, there is never a correct system configuration. This source of downtime arises from lack of *alignment*;

3. **Intrinsic reliability:** the intrinsic reliability of the components that make up the system, the way in which they are connected; and their dependency on operational level agreements in place for break/fix constitute the usual domain of availability analysis and typically the focus of SLAs in current IT Outsourcing (ITO) and application management contracts;

4. **Network:** a catch all for lack of availability caused by the network itself being swamped by non authorised, non business related, traffic perhaps due to a Denial of Service (DoS) or worm attack. This source of downtime arises from *vulnerabilities*.

We make an essential distinction between downtime arising from vulnerabilities (1, 4) and alignment (2). Typical analyses of threat environments and attack profiles have tended to concentrate on the former (Wing et al. 2002). Both, however, must be monetized for any derived SLA to be considered meaningful.

3.1. A description of the model

The parameters for the Demos2k model corresponding to Figure 2 are given in **Appendix B** structured according to the components listed in **Appendix A**. The following internal processes consume security operations staff resources:

1. Continuous change management (relationship between users, resources and processes) which has been aggregated into a single business alignment services process;

2. Repair compromised or damaged machines;

3. Processes associated with patch management including testing and deployment;

4. Implementing workarounds as mitigating actions;

5. Deployment of IPS signatures;

6. Vulnerability assessment — determining corporate patching deadlines and associated risk levels.

The threat model provides the stimulus to all of the security operations tasks indicated above. We acknowledge that it is a radical simplification of the known threat space — for example, the vulnerability intelligence capability could be split into White Hat, Grey Hat and Black Hat operations — but the present model suffices for our present purposes. It is clearly an area of modelling that warrants further

---

[2] We have used business (alignment, process, etc.) throughout and intend this to be a generic label for any enterprise.
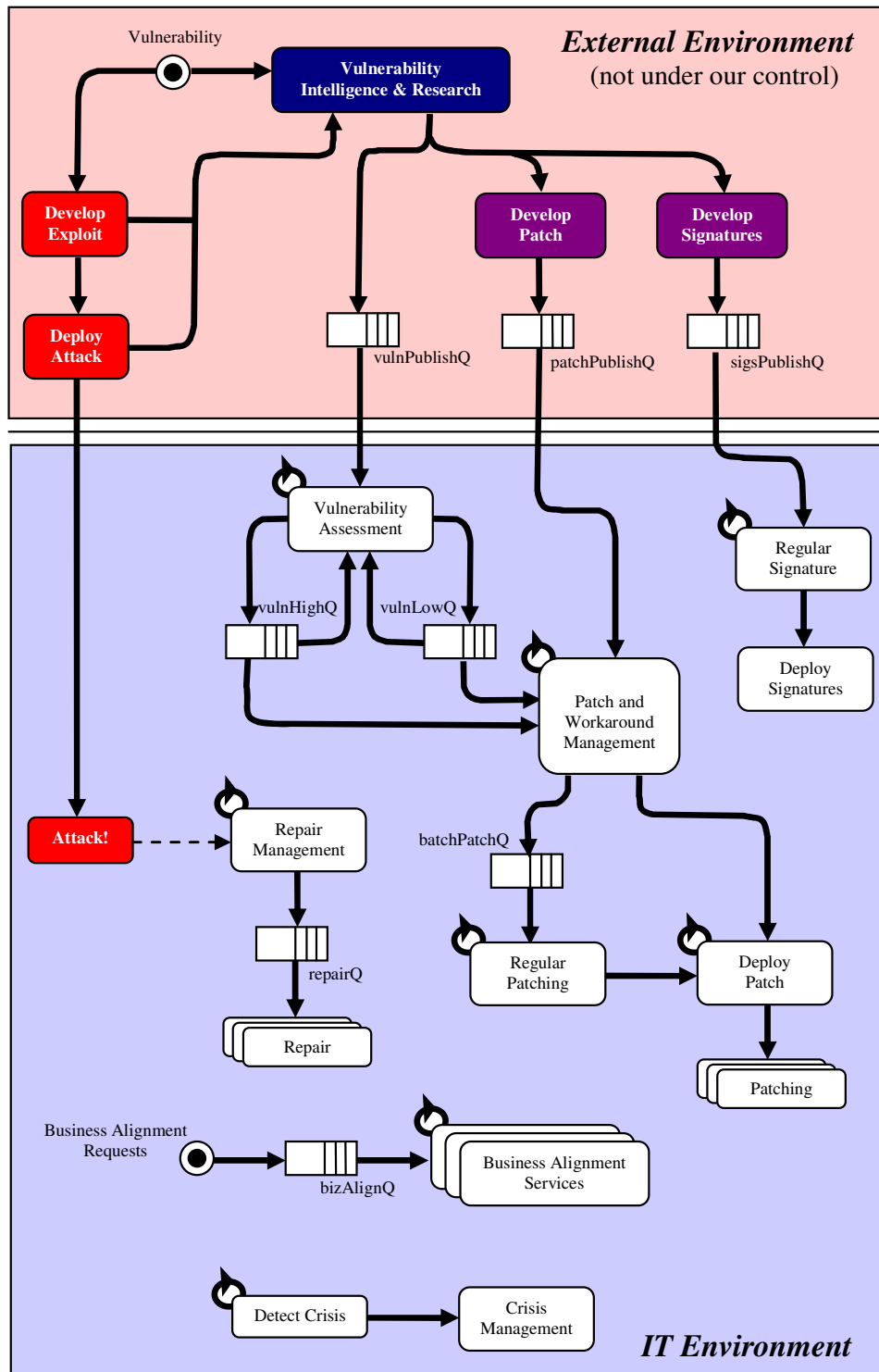
Figure 2. High-level process view of causative spawning dependencies in the model. The thick arrows represent event flows between processes and queues. Processes decorated with circular arrows indicate 'perpetual' processes that continually execute, consuming events as they do so.

11

development in its own right and is potentially useful outside the context of this paper; however we have made the following assumptions:

All vulnerabilities are naturally present in all *unpatched* systems — they do not need to be 'caught' or 'infected':

- Viruses, Worms, and Trojans (VWT) exploit vulnerabilities that remain unpatched. These VWT are the *infection agents*;

- Typically, there has to be some external access for VWT to become active;

- Patching systems will eliminate the vulnerabilities *and* remove the effects of the infective agents.

In order to avoid situations where our model suffers from the runaway affects of an attack leading to an implausible number of unavailable machines, we have modelled spread effects by introducing forms of damping. The number of machines requiring repair and the number of machines on the repair queue will grow under attack at two different rates, where the time constant of the repair queue is slower than the number of machines requiring repair. Since the length of the repair queue would be monitored we introduce a 'crisis' threshold. At the time the crisis occurs the following take place:

- The organization is effectively down for a number of days;

- Utilization of security operations staff during this down time is 100%; and

- At the end of the down time the number of machines on the repair queue becomes equal to the number of machines that needed repair at the time of the crisis.

The present model can be broadly calibrated by measuring the following aggregated parameters from the organization:

- The aggregation of individual FTE security operations staff into teams that govern the concurrency limits within the model;

- The average amount of time taken to undertake the processes and the number of security operations staff typically assigned to work on each process.

The following is a suggested list of metrics. Not all of these may be measurable in practice but are included here for discussion.

1. The amount of time taken to assess the potential impact of a new vulnerability for which a patch exists before deciding to deploy;

2. The effectiveness of attacks, patching and workarounds;

3. Deployment coverage measurements associated with patch management, Virus signature distribution and workarounds;

4. The characteristics of specific attacks;

5. The characteristics of the patching process.

5. Results

The results from the model presented here should be considered as preliminary, demonstrating the basic feasibility of the approach rather than a specific prediction for a specific system. We also assume that should an organization decide to use an availability SLA then loss of availability can be adequately monetized. The model is currently homogenous with respect to systems but the approach we have used could be replicated to represent different processes for different subsystems but still drawing from a shared

resource pool and with no loss of desirable properties; parametrizing Demos2k processes through use of a *job ticket* mechanism is reasonably easy to code.

The model has been used to investigate two key questions: firstly, the impact on predicted availability of a system and utilization of reducing the number of security operations staff resources; and, secondly, the impact on availability and utilization if the threat environment changes. Some of the runs produced using an early forerunner of this model produced what could only be regarded as devastating loss of availability due to single attack, effectively rendering the organization non-viable. This early model was clearly too naïve and was rectified by the creation of a more elaborate 'damped attack' model described in §3.1 which also permitted greater subtlety in the modelling of defensive processes.

We have chosen a system with 20,000 devices which equates to a high end Small to Medium Enterprise (SME) or institution similar in size to a medium-sized European University. Given that the parameters in the model have not been calibrated we chose a 'reasonable' starting set and explored the number of security operations staff that gave achievable utilization rates when exploring increased threat and reduced numbers of operational staff. This gave a baseline model in which 3 security operations FTE staff were required, *assuming single site operation*. In order to acquire reasonable statistics, each simulation run of 365 days was repeated 100 times. Using the model as shown in Appendix B as a baseline we obtain an average predicted availability of 98.15%. The corresponding average utilization of the security operations staff was 57.45%. Data from a typical run with $N_p=0$ showing daily availability over a period of 365 days is shown in Figure 3 below.
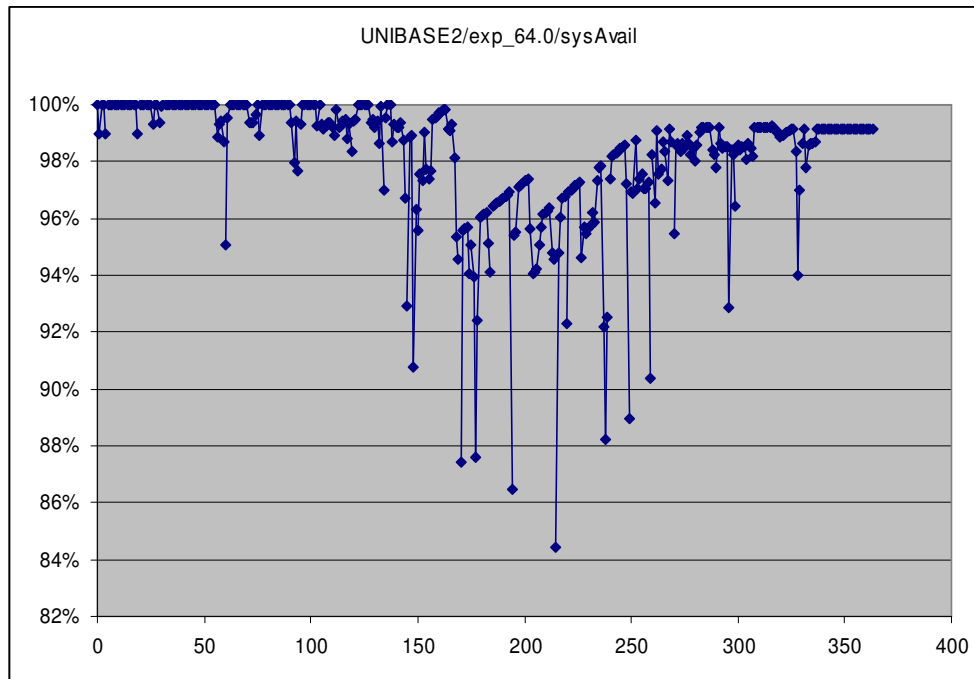
Figure 3. Daily availability per day, from a typical simulation run. This shows the many minor fluctuations on availability caused by a continuous stream of change management requests arising from misalignment. In addition, there are more significant dips arising from attacks enabled by the presence of vulnerabilities etc.

The model presented here is quite complex for Demos2k and presents some challenges in understanding the complexity of the dynamic behaviour exhibited.

Data visualization

We have used a 3D visualization approach built by one of the Demos2k authors that renders some or all of a trace file from an experimental run into VRML. In the illustration given in Figure 4 below, various data traces spanning a whole year are shown as labelled columns. The various planes (shown in grey) represent points in time (days) cutting across each column. The particular traces shown here are the `needsRepairs` and `needsPatching` queues, the cumulative `cAttackDeployed` and `cPatchDev` counts and the `attackImpact` and `sysAvail` measures. The VRML visualization gives the analyst modeller an intuitive way of viewing and selecting data to highlight aspects of interest arising in Demos2k traces. As such, only a subset of the data that could be visualized from the Demos2k traces generated is given here.
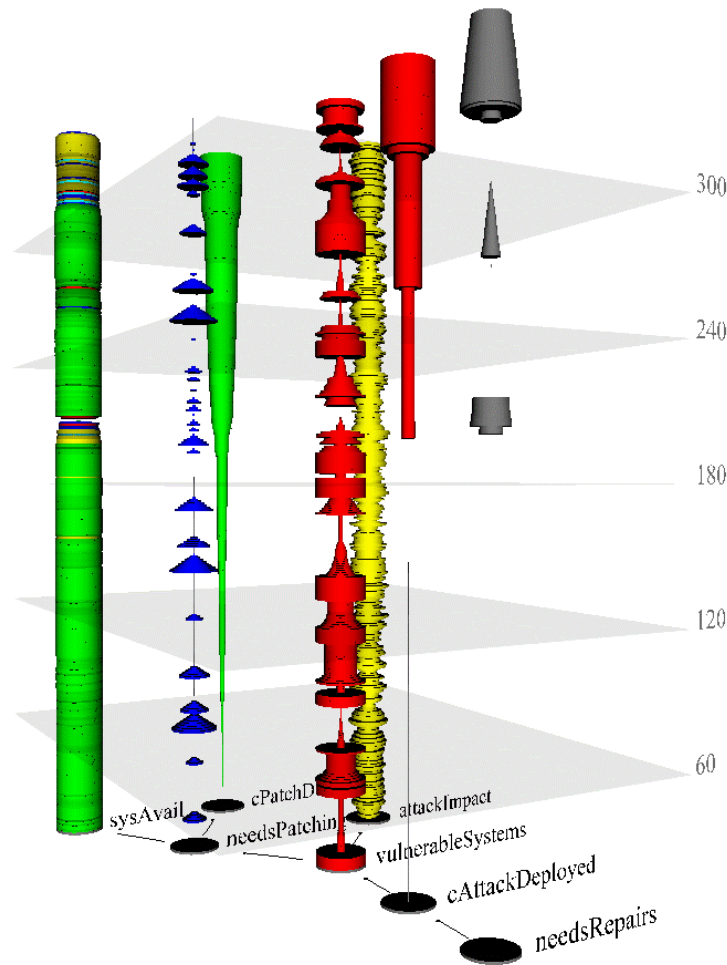
Figure 4. Visualization using VRML of data traces selected from an experimental run of the model

Data summary

With all other parameters held constant, the effect of reducing the number of operations staff by 33% and of increasing the rate at which vulnerabilities are discovered by 50%, have been investigated and summarized in Table 1. In terms of meeting the requirement of achieving cost-effective modelling, each 365 day simulation takes approximately 120 seconds to run on a 1.6GHz Pentium notebook machine. A more thorough examination of the space of models provides an interesting topic for further work.

14

	N_{ops}	Vulnerability Rate	Availability (Average)	Availability (Min)	Availability (Max)	Utilization (Average)	Utilization (Min)	Utilization (Max)	$P(N_p=0)$	$P(N_p=1)$	$P(N_p=2)$	$P(N_{pa}>2)$
1	3	Negexp(365/24)	98.15%	95.86%	99.54%	57.45%	42.35%	81.96%	86%	13%	1%	0%
	N_{ops}	Vulnerability Rate	Availability (Average)	Availability (Min)	Availability (Max)	Utilization (Average)	Utilization (Min)	Utilization (Max)	$P(N_p=0)$	$P(N_p=1)$	$P(N_p=2)$	$P(N_p>2)$
2	0%	+50%	-0.95%	-3.05%	-0.24%	+7.55%	+5.09%	+7.38%	71%	28%	1%	0%
3	-33%	0%	-0.82%	-2.47%	-0.31%	+22.38%	+22.93%	+16.47%	72%	25%	3%	0%
4	-33%	+50%	-1.98%	-3.42%	-0.39%	+26.34%	+24.48%	+14.51%	50%	40%	10%	0%

Table 1. Summary of data for 4 scenarios: changing the number of operations staff and the rate of discovery of vulnerabilities. The data in the first row are presented as a baseline case. Data for availability and utilization for cases 2, 3 and 4 are presented as deltas.

6. Discussion and conclusions

In this paper, we show early development stages of a framework within which meaningful security SLAs for security operations can be developed. We have outlined a mathematically well-founded modelling system, partially realized within the Demos2k tool, allowing us to predict the availability of systems vulnerable to downtime arising from attacks and from business misalignments. The attacks in question are assumed to exploit the pervasive presence of systems security vulnerabilities.

Based on our current understanding of security operations dynamics, the results obtained from simulating the model show that there is not an unexpected correlation between increases in utilization rate for security operations staff and reductions in number of operations staff available and similarly, increases in potent threat from the environment.

It is clear that the quality and significance of our modelling strongly depends upon the structural description of business processes we obtained through using it. This means that we can critique and test how well our understanding is reflected by the process structure. A further strength of our process-driven modelling approach is that our models happen to be executable and can thus generate synthetic data for analysis. This potentially allows us to make quantitative, predictive conjectures that are measurable and hence testable, under the appropriate circumstances.

We have thus successfully modelled (lack of) availability that arises from attacks and business misalignment. This demonstrates a rational basis for answering some of the basic questions posed earlier and fulfils the basic requirement of providing a predictive capability for a candidate availability SLA, modulo calibration with actual business situations. Such a capability clearly contributes to helping make investment decisions with respect to security operations and the associated security infrastructure.

There are a number of ways to extend the reach and scope of the modelling work reported here; for example:

- Exploring how much additional detail to include in the model to obtain calibration with particular business instances. There is an important trade-off to be made here since adding too much detail and the model becomes intractable and unanalysable — too little detail and the model becomes too broad and much harder to correlate with specific business process characteristics;

- We encountered various runaway conditions in the model that were directly provoked by attacks, illustrating a need for more detailed models of attack mechanisms and defensive processes. The point here is that although any given attack may not in itself be all that damaging, but given how stretched the allocation of resources could be, the attack may be "the straw that breaks the camel's back", forcing a *crisis* to occur. Future versions of the model could explore these issues of *sensitivity*, given greater depth of attack/threat modelling;

- It is clear from our model that a number of processes (e.g., patching) appear to be critical to the effective network defence of a large organization. How critical are they really? What would happen if they could be replaced by less expensive options? Adding extra automation can reduce the need for staff — but what happens when these complex systems breakdown (as they inevitably will do). What about maintenance schedules? What are the cost-benefit trade-offs?

- The modelling framework needs to incorporate *location-sensitive* resources to fully account for the richness inherent in many business situations. For example, this would permit us to model separations better; the use of an operations centre that is geographically situated elsewhere implies that there may be greater potential for communications disruption, differences in staffing allocation, increased dependence upon the reliability of networking infrastructure, etc. All of these would have cost implications for the kinds of resources needed;

- Our models explicitly cover availability issues for security operations. It would be interesting to extend our approach to cover both confidentiality and integrity issues as well, so obtaining a well-rounded approach to understanding the economic cost/benefits of IT security.

7. Acknowledgements

References

Abadi, M., Burrows, M., Lampson, B., Plotkin, G., A calculus for access control in distributed systems. *ACM Trans. Prog. Lang. Sys.* 15(4):706 – 734, 1993.

Anderson, R., 2001. Why Information Security is Hard — An Economic Perspective. *Proc. 17th. Annual Computer Security Applications Conference*. Available at http://www.acsac.org.

Birtwistle, G., Tofts, C., The Operational Semantics of Demos: part I, *Transactions of the Simulation Society* 10(4):299 – 333, 1993.

Birtwistle, G., Tofts, C., The Operational Semantics of Demos: part II, *Transactions of the Simulation Society*, 11(4):303 – 337, 1995.

Christodolou A., Taylor R., and C. Tofts, 2000. Demos 2000. http://www.demos2k.org .

Galmiche, D. , D. Méry, and D. Pym, 2005. The Semantics of BI and Resource Tableaux. *Math. Struct. Comp. Sci.*. 15:1033-1088., 2005

Gordon, L., Loeb, M., 2006. *Managing Cybersecurity Resources: A Financial Perspective*. New York. McGraw Hill.

HP 2006 *A Bright Idea.* *http*://gem.compaq.com/gemstore/sites/state/emag2/pdfs/Winter2006/03_HPLabs.pdf

Milner, R., 1983. Calculi for Synchrony and Asynchrony. *Theoretical Computer Science* 25(3), 267-310, 1983.

Milner, R., 1989. *Communication and Concurrency*. Prentice Hall.

Monahan, B., Pym, D.,2006. A Structural and Stochastic Modelling Philosophy for Systems Integrity. Technical Report, http://www.hpl.hp.com/techreports/2006/HPL-2006-35.html

O'Hearn, P. and D. Pym, 1999. The Logic of Bunched Implications. *Bulletin of Symbolic Logic* 5(2), 1999, 215-243.

Pym, D., 1999. On Bunched Predicate Logic. Proc. LICS '99, 183-192, IEEE Computer Society Press.

Pym, D., 2002. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series 26, Kluwer Academic Publishers.

Pym, D. and C. Tofts, 2006. A Calculus and Logic of Resources and Processes. To appear, *Formal Aspects of Computing*. Preliminary version available at http://www.hpl.hp.com/techreports/2004/HPL-2004-170R1.html

Stirling, C., 2001. *Modal and Temporal Properties of Processes*. Springer, 2001.

Sheyner, O., Haines, J., Jha, S., Lippmann, R., and J.M.Wing, 2002, Automated Generation and Analysis of Attack Graphs*, Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA.

Tofts, C. 2003, *Efficiently modelling resource in a process algebra*. Technical Report HPL-2003-181, HP Laboratories, Bristol.

Appendix A: A Brief Guide to Demos2k

In reality, Demos2k is two things — firstly, it is a *semantically justified* discrete-event systems modelling language (see §3.1); and, secondly, it is a simulation-based environment to support the examination and exploration of systems so described.

An advantage of this semantic justification activity is that, having done this work, it contributes extensively to the integrity of the implementation, thus ensuring *accuracy* and *fidelity* of the simulation results obtained using the tools. Broadly speaking, this means that Demos2k users can be highly confident in the numbers produced during and resulting from their simulations and in the patterns of behaviour observed. With this underpinning, we can be much more certain that our results are genuine *consequences* of the model and not mere artefacts of simulation.

The Demos2k environment has been designed to support the precise examination of simulation oriented descriptions of systems. These can be compiled or automatically rewritten into multiple representations dependent upon the questions that must be asked of the model such as correctness, performance, availability, or agility, etc.

Systems descriptions written in Demos2k tend to be high-level, pleasingly short and to the point. The modelling philosophy supported is very much akin to 'extreme modelling', where the systems analyst/modeller can rapidly construct high-level models representing the customer's core business concerns. A key contribution to this capability is the exploitation of probability theory to abstract away from extraneous details.

We now outline the basic 'shape' of a typical Demos2k definition of a model. Although not syntactically mandated in any sense, as a general rule Demos2k models/programs pragmatically adopt the following pattern:

Constant definitions:

- Demos2k constants are special in that they may be defined in terms of probability distributions – each time such 'constants' are evaluated during simulation, a fresh sample is taken from the specified distribution. The probability distributions supported include standard distributions such as Uniform, Binomial, Geometric, Negative Exponential, Normal, Poisson, and Weibull, as well as arbitrary point/discrete distributions;

Global variable definitions:

- Variables are typically used to count the number of events of a certain kind, or the number of times a particular process is activated;

Resource definitions:

- Resources represent pure synchronizations (in the process-calculus sense) and can be claimed and released by means of `getR` and `putR` expressions;

Bin definitions:

- Bins represent synchronizable entities (note that the term 'resource' is used in the rest of the paper to encompass both the Demos2k notion of 'resource' and the Demos2k notion of 'bin', as described here) into which some quantity of material may be placed and retrieved. These may be used to provide the effect of one entity making a synchronous, concurrent process call on another;

Class definitions:

- Each entity is a concurrently executing instance of some class. Classes thus represent the behaviour of entities in conventional procedural terms, by manipulating resources in some fashion and by 'holding' (letting time pass) for defined periods of time;

Initial model population, and entity creation;

Run length control, typically a `hold` of some fixed duration;

The all-important `close` statement ends the simulation run.

In this form, we may regard Demos2k descriptions as defining behaviour in terms of a Dijkstra-like guarded command language. All active commands test the current system state. If the condition they represent can be met then they are executed --- otherwise they are blocked until such time as the condition holds, if at all. Note that Demos2k simulations will typically run for a specified length of time. If either deadlock or livelock arise during simulation runs then these situations are checked for pragmatically. The major difference between process oriented simulation languages (like Demos2k) and pure guarded command languages is that the conditions may have side effects, principally due to the assignment of resource to the active entity. Hence change of state is mediated not only by assignments to variables, and by the claim of resource, but also by entities becoming resources themselves.

Demos2k has been given a simple, elegant and informative semantics, abstracting away from the stochastic data collection, in the process calculi SCCS and CCS (Milner 1983, Milner 1989). It can be argued that the representation of resource in the synchronous semantics (SCCS) is superior to that in the asynchronous semantics (CCS) - see (Tofts 2003).

Appendix B contains a condensed version of the security operations model discussed in §3.1 of this paper – a complete Demos2k model may be found in the appendix to (Monahan and Pym 2006).

19

Appendix B – Availability Model

```
// Security operations model – Mike Yearworth, Brian Monahan & David Pym
// version 1.5 (condensed)
// 19 July 2006 – update a
// Constants
// Timescale constants
cons days                   = 1;
cons hrs                    = days/24;
cons mins                   = hrs/60;
cons secs                   = mins/60;
// Simulation time
cons holdTime               = 365;                      // int: sim. period -- in days
// General Parameters
cons totalSys               = 20000;                    // number of desktop systems under management
cons NsecOps                = 2;                        // number of operations staff
cons Nanalysts              = 1;                        // number of security risk analysts
cons patchTeams             = 4;                        // number of patch teams
cons repairStreams          = 4;                        // number of repair streams
cons serviceTeams           = 4;                        // number of biz align service teams
cons sysForAlign            = 1;                        // number of systems needed per alignment request
cons maxPatchLimit          = 30;                       // maximum number of systems patched concurrently by single team
cons lowerAvailLimit        = 90/100;                   // lower limit of acceptable availability
cons max_days_unavailable_limit = 3;                    // maximum number of days consecutively unacceptable availability
cons max_repairs_limit      = 850;                      // max acceptable length of repair queue
cons clearUpTime            = 2;                        // number of days needed to clear up crisis
cons accountingSamplesPerDay = (24*(60/15));            // number of accounting samples taken per day
cons rebootTime             = normal(10*mins, 1*mins);  // avg. reboot time
cons patchDeployTime        = normal(1*hrs, 15*mins);   // avg. time to apply a patch to system
cons sigsDeployTime         = normal(10*mins, 1*mins);  // avg. time to update signatures
// Environment related
cons vulnerabilityInterval  = negexp(365/24);           // Avg. time between vulnerability discovery
cons isExploitable          = binom(1, 40/100);         // Prob. of vuln. being exploitable
cons devExploitTime         = negexp(19*days);          // Avg. time to develop exploit
cons vulnDiscoveryTime      = negexp(14*days);          // Avg. time to discover/expose intell. on vuln.
cons sigsEffectiveness      = uniform(10/100, 20/100);  // level of effectiveness of signatures
cons attackEffectiveness    = uniform(05/100, 85/100);  // normalized measure of attack capability ...
cons attackDeploymentTime   = negexp(20*days);          // Avg. time to deploy an attack, given an exploit exists
```

```
cons isZeroDayAttack          = binom(1, 1/10);           // Prob. of exploit being immediately deployable
cons isDeployableAttack       = binom(1, 8/10);           // Prob. of exploit being eventually deployable
cons isExposable              = binom(1, 9/10);           // Prob. of vulnerability being exposable (i.e. subject of intell.)
cons patchDevTime             = negexp(15*days);          // Avg. time to develop patch
cons sigDevTime               = negexp(5*days);           // Avg. time to develop signatures
cons attackRounds             = puni(1, 3);               // Avg. number of attack attempts.
cons attackInterval           = normal(2*days, 16*hrs);   // Avg. time between attack attempts
// Business related staffing limits
cons staffForPatching         = 1;                        // number of staff needed for patching activity
cons staffForSigs             = 1;                        // number of staff needed for updating signatures
cons staffForAlign            = 1;                        // number of staff needed for alignment request
cons staffForRepair           = 1;                        // number of staff needed for repairing systems
// Biz alignment
cons bizAlignInterval         = negexp((1/30)*days);      // avg. time between requests
cons bizAlignTime             = negexp(60*mins);          // avg. time taken to complete biz align tasks.
cons someReassessment         = puni(0, 3);               // avg. number of vulnerability reassessments performed
cons vulnAssessTime           = negexp(2*hrs);            // avg. time taken to assess vulnerabilities
cons patchAssessTime          = negexp(2*hrs);            // avg. time taken to assess suitability of patches
cons assessmentInterval       = negexp(1*days);           // avg. time between vulnerability assessments
cons isVulnHigh               = binom(1, 2/10);           // prob of vulnerability being urgent
cons isVulnLow                = binom(1,  7/10);          // prob of vulnerability being non-urgent, but useful
cons repairQCheckInTime       = normal(8*mins, 1*mins);   // time taken to check in systems to repairQ
cons canQuickFix              = binom(1, 1/10);           // prob. of system being fixable quickly
cons repairTime               = normal(3*hrs, 10*mins);   // avg. repair time
cons patch_is_irrelevant      = binom(1, 1/10);           // prob. of patch being neither relevant nor useful
cons patchMaintenanceInterval = normal(14*days, 1*days);  // avg. time between patch maintenance
cons sigMaintenanceInterval   = normal(7*days,  1*days);  // avg. time between sig. defence maintenance
cons systemsNeedingPatching   = uniform(10/100, 95/100);  // proportion of systems that need patching
cons sysNeedSigs              = uniform(50/100, 90/100);  // proportion of systems requiring signatures
// Variables
var day                       = 0;
var vulnerableSystems         = 0;
var attackImpact              = 0;
var sysAvail                  = 1;
var cSysAvail                 = 0;
var cStaffUtil                = 0;
var availSys                  = totalSys;
var needsRepairs              = 0;
var needsPatching             = 0;
```

```
var online                        = 1;
var daysUnavailable               = 0;
var manpowerUsedToday             = 0;
var systemsAvailToday             = 0;
var cVuln                         = 0;
var cCrises                       = 0;
var cAttacked                     = 0;
var cRepaired                     = 0;
var cExploit                      = 0;
var cPatchDev                     = 0;
var cSigDev                       = 0;
var cAttackDeployed               = 0;
var cBizRequests                  = 0;
var cBizRequestsServed            = 0;
var cPatchesApplied               = 0;
var cSigsApplied                  = 0;
// Resources
res(lock, 1);
res(analysts, Nanalysts);
res(opsStaff, NsecOps);
bin(vulnAssessQ, 0);
bin(patchPublishQ, 0);
bin(sigPublishQ, 0);
bin(bizAlignQ, 0);
bin(vulnHighQ, 0);
bin(vulnLowQ, 0);
bin(batchPatchQ, 0);
bin(repairQ, 0);
// Classes
// External processes/activities
// (the { . . . } notation below represents details omitted for lack of space)
class vulnerable = { . . . }
class devExploit = { . . . }
class vulnIntelligence = { . . . }
class devPatch = { . . . }
class devSig = { . . . }
class deployAttack = { . . . }
class attack = { . . . }
// Internal IT activities
```

```
class bizAlignRequests = { . . . }
class bizAlignService = { . . . }
class vulnerabilityAssessment = { . . . }
class assessment = { . . . }
class patchManagement = { . . . }
class maintainPatch = { . . . }
class deployPatch = { . . . }
class patchApplication = { . . . }
class deploySig = { . . . }
class repairManagement = { . . . }
class repair = { . . . }
class detectCrisis = { . . . }
class crisisManagement = { . . . }
// Management
class accounting = = { . . . }
class reporting = { . . . }
// Entities
entity(vuln, vulnerable, vulnerabilityInterval);       // generate vulnerabilities ...
entity(biz,  bizAlignRequests, 0);                     // generate biz alignment requests ...
entity(vulnAssess, vulnerabilityAssessment, 0);        // vulnerability assessment ...
entity(patchMgmt, patchManagement, 0);                 // patch management ...
entity(patchMtn, maintainPatch, 0);                    // maintain patching process ...
entity(sigDefMtn, maintainSigDefence, 0);              // maintain signature defence process ...
entity(patchDep, deployPatch, 0);                      // deploy patches ...
entity(sigDefDep, deploySig, 0);                       // deploy defensive signatures ...
entity(repairMgmt, repairManagement, 0);               // repair Management ...
do serviceTeams  { entity(bizService,  bizAlignService, 0); }   // service biz alignment requests ...
do repairStreams { entity(repair,      repair,       0); }      // repair teams ...
// business mgmt & reporting
entity(check, detectCrisis, 0);                        // check for crisis conditions
entity(accounts, accounting, 0);                       // perform accounting ...
entity(reports, reporting, 0);                         // perform reporting ...
// Run the simulation ...
hold(holdTime);
close;
```

23