A Structural and Stochastic Modelling Philosophy for Systems Integrity

Brian Monahan, David Pym
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2006-35
February 27, 2006*

philosophy, modelling, logic, resource semantics, probability, queuing theory, location, performance, security, access control, authorization, utility computing services, demos, stewardship

We present an essentially philosophical account of a mathematical approach to systems and services modelling for the purpose for understanding the functionality, performance, reliability, and security - in short, the integrity - of ICT systems. We describe the economic background to the need for effective modelling technologies, and explain the principal strengths of a key existing technology, the experience of which we build upon. Building squarely on this industrially validated practical experience, we then describe a rather general but directly applicable mathematical framework, and discuss how to model central issues in access control, such as rôles and impersonation, data integrity, and most interestingly, stewardship. Our mathematical framework combines combinatorial, logical, algebraic, topological, and, critically, stochastic methods. We emphasize that we are not overly concerned with the formal specification of the detailed behaviour of systems and services. Rather, our interest is focused upon a framework for building particular mathematical models of specific aspects of enterprise-scale systems and services at appropriate levels of abstraction. This framework is constructed to help explore questions concerning, for instance, combinations of services availability and systems accessibility properties. In particular, we aim to use models that capture performance to address also systems security questions such as whether a given system model is capable of complying with a security policy requirement, as expressed by a service-level agreement. Further, we aim to quantify the operational impact and cost of both failure to comply and of transition to compliance.

* Internal Accession Date Only
Approved for External Publication

A Structural and Stochastic Modelling Philosophy for Systems Integrity

Brian Monahan and David Pym

Hewlett-Packard Laboratories
Bristol, U.K.
{brian.monahan, david.pym}@hp.com

Abstract. We present an essentially philosophical account of a mathematical approach to systems and services modelling for the purpose for understanding the functionality, performance, reliability, and security — in short, the integrity — of ICT systems. We describe the economic background to the need for effective modelling technologies, and explain the principal strengths of a key existing technology, the experience of which we build upon. Building squarely on this industrially validated practical experience, we then describe a rather general but directly applicable mathematical framework, and discuss how to model central issues in access control, such as rôles and impersonation, data integrity, and most interestingly, stewardship. Our mathematical framework combines combinatorial, logical, algebraic, topological, and, critically, stochastic methods. We emphasize that we are not overly concerned with the formal specification of the detailed behaviour of systems and services. Rather, our interest is focused upon a framework for building particular mathematical models of specific aspects of enterprise-scale systems and services at appropriate levels of abstraction. This framework is constructed to help explore questions concerning, for instance, combinations of services availability and systems accessibility properties. In particular, we aim to use models that capture performance to address also systems security questions such as whether a given system model is capable of complying with a security policy requirement, as expressed by a service-level agreement. Further, we aim to quantify the operational impact and cost of both failure to comply and of transition to compliance.

1 A Modelling Philosophy

Failures of large systems projects to meet upon delivery the expectations of their users and even their designers in respect of function, performance, reliability, and security — in short, integrity — are widespread and familiar problems that can entail substantial direct costs and significant under-performance in the client organizations. Many reasons for these failures are commonly identified including, for example, apparently overwhelming complexity, inadequate description or understanding of system specifications, failure to reuse appropriate proven solutions, poor communication of the experience of failure, and inadequate availability of education, skills, and standards in systems understanding.

We believe that, in order to address this situation, there is a need for a systematic, rigorous, yet tractable, framework for understanding how systems design decisions affect systems integrity. In this paper, we describe how a modelling approach based on

process calculus and logic can be used as the basis for such a framework. We stress that although our framework is based on ideas from process calculus and logic, it is not intended as a basis for formally specifying systems. Rather, it is intended to support the construction of models constructed at levels of abstraction that are appropriate to answering specific questions about the properties or behaviour of the system and the services it supports — considered and modelled as processes — in specific circumstances. To this end, we draw our inspiration from the philosophy and experience of the work of Birtwistle [5] and others, using the **Demos 2000** modelling system, for capturing large-scale systems in industrially significant contexts. This work has led to the following conclusions about the value of such a modelling approach [42]:

- The act of creating a model forces an organization to consider and review the structure of the business, investment or product that they are proposing to create;
- The model acts as important part of the documentation of a system; the evolution of such a model, if documented, is an invaluable aid in the audit of a project;
- The model can act as a valuable communications aid, allowing discussions to be grounded in a common representation;
- Models allow for rapid exploration of the decision space that an organization is operating in, enabling multiple scenarios to be played out at low risk;
- Models may be used to qualify and then check real systems;
- Models can demonstrate the sensitivity of a system to environmental changes, enabling users to design out (as much as is possible) potentially disruptive nonlinearities in the system behaviour;
- Models can be used to check the correctness of approaches to problem solving;
- Models permit the early capture of error, as they permit nonexistent systems to be studied, with the well known benefit of capture time against value saved.

We will thus make some use of **Demos 2000** (www.demos2k.org), a semantically justified [7, 8] discrete-event systems modelling language, to concretely illustrate our approach in the example presented later in § 3. **Demos 2000** itself has a long-standing pedigree having been originally derived from Graham Birstwistle's work [5]. The essential components of **Demos 2000** are the following:

- Captures the behaviour of *entities* in terms of actions involving the manipulation of *resources*;
- Captures systems of queues in terms of resources and synchronized 'bins';
- Provides a stochastic representation of events that must be handled by the model.

We present a compact, conceptual summary of **Demos 2000** in Appendix A.

Inspired by these observations, we describe, in § 2, a calculus **SCRP** of *resource-processes* due to Pym and Tofts [37] based, on the one hand, on a simple calculus of synchronous processes in the style of SCCS and, on the other, on the *resource semantics* introduced by Pym and O'Hearn [36, 38, 39]. The calculus uses an explicit representation of resource and models the co-evolution of resources and processes with synchronization constrained by the availability of resources. Using extensions of ideas from Pym and O'Hearn's bunched logic, **BI**, a modal logic, **MBI**, can be used to give a characterization of bisimulation for **SCRP**, analogously to Hennessy-Milner logic's characterization of bisimulation in CCS [29]. This characterization is compositional in the concurrent and local structure of systems.

1.1 ICT Security

ICT *security*, however, is no longer regarded as a pure technology issue, at least within the sphere of corporate business. It has become a truism to say that ICT security is a process — something that happens to create smooth operating conditions for business. Modern corporate management makes essential use of metrics — numerical measures — to demonstrate business performance and hence show impact upon shareholder value. Questions then arise of how to estimate the level of *security performance* that these security-related processes achieve and, indeed, how to specify what their goals should be and how to predict outcomes and impacts.

Another standard corporate management technique is the use of explicit Service Level Agreements (SLA) between the internal functional units of an organization and, more typically, with external services providers. These SLAs provide a contractual statement for what is to delivered to the organization by the provider, internal or external. With respect to security, these contractual statements have commonly taken the form of pure *policy compliance* statements e.g. conformance to ISO17799 and such like. These requirements are then audited on a regular basis, the results of which may contribute to a corporate annual report, and thus have impact upon business confidence.

As we see it, there is an increasing trend for organizational security policy to extend beyond passive compliance against prevailing best-practice standards towards compliance with security policies that inherently involve SLA-style performance goals to be met. We strongly believe that security concerns need to be addressed within an economic performance and process modelling framework such as that presented here.

Security is mostly concerned with risk management, risk reduction and mitigation. Risk management is a necessary approach to security risk, as opposed to 'risk avoidance', since security issues cannot be evaded with any confidence in today's network-enabled global business environment.

The benefit accruing from security lies in the reduced disruption to the functioning of a business or organization. Security is therefore context-dependent and thus involves a notion of *scope of control*, an abstract form of location. In particular, security is relative to the potential threats and risks to be defended against. This means that we are interested in capturing what these risks are and estimating the costs of mitigation failure. Naturally, such costs have to be weighed against the cost of providing such mitigation in the first place. The earlier attempt to semi-formally model aspects of systems security presented in [31] alluded to some of the characteristics we explore here: e.g. the benefit of security as lack of disruption, and a concern for distributed access control.

Thus, the economic value of security in cost-benefit terms [19] is principally due to the cost savings arising from preventing disruption and improving reliability and trust, thus creating smooth operating conditions for business transactions. But security engineering is often at odds with reliability engineering. For example, replication and mirroring are standard strategies for increasing the reliability of a system. Unfortunately, at the same time, making multiple copies available increases the number of points at which confidentiality could be breached. Thus, a naïve approach to improving availability may have serious engineering implications for the appropriate maintenance of confidentiality. The challenge is then how to provide and engineer reliable systems and services that are at the same time secure.

1.2 Utility computing

Business is constantly seeking ways and means to improve the cost-effectiveness of ICT systems and their Return On ICT Investment. One approach to doing this is the so-called 'Utility Computing' model, in which companies can 'rent' ICT capability in a flexible and adaptive manner [32]. Such capability is typically delivered to the customer from a highly automated data center environment over high-speed switched networks. Organisations that provide this capability are known as Utility Computing Service Providers (UCSPs). The term 'utility computing' derives from the analogy with standard utility services such as water, gas and electricity.

Naturally, to provide utility computing to customers economically, UCSPs will typically need to operate a computing environment that is shared across all their customers. Furthermore, from each customers point of view, they should only be aware of those resources that appear to have been allocated to them. Each customer should confidently expect to work as though they have *exclusive* access to their resources, even though this will hardly ever be true. In many ways, this situation is strongly analogous to the separation requirements for a multi-user mainframe system — except that in this case, the system is not a single machine but instead many machines, networked together.

To make effective use of the utility computing capabilities they have rented, however, customers will typically need to combine these capabilities with access to their own highly-valued information assets, so opening up a possible route to their malicious compromise. Accordingly, the sharing implicit in the utility computing model represents a considerable risk of exposure and compromise to the customer's assets.

Thus the challenge for any UCSP is to implement a flexible, shared, and secure computing infrastructure in such a way that their customers can safely use the (aggregated) resources they have been allocated, without concern for the activities of other customers or the operations staff also using the networks. It is clear that effective security is a necessary and fundamental requirement for the success of utility computing.

1.3 Extending Performance Modelling Towards Access Control

Building on our mathematical framework for performance modelling, as exemplified in a simple form by **Demos 2000**, we extend our framework to account for a key aspect of security, namely access control. We begin, in § 3, with our conceptual view of the problem, supported by a **Demos 2000** model. We then discuss, in § 4, how to extend the **SCRP-MBI** framework to provide an appropriate mathematical framework. The key idea is that of *location*. We adopt a modelling approach to understanding location, identifying the key axiomatic properties of the notion — sublocations, substitutions, connections, and products — and given a useful leading example — based on directed graphs. We explain how to reconstruct the basic components of the analysis of access control given in [2, 25] in our setting, making certain concepts precise.

Before proceeding to our main development, it is necessary to explain the central rôle of stochastic methods in our modelling philosophy. Probability theory is very widely used in cryptography, typically to determine the likelihood of effective attacks on ciphers and their keys. Many security definitions are stated in terms of probabilistic concepts, as the following typical example shows: Consider a family of cipher-cracking

problems involving some numerical security parameter, k. A cryptographic attack on the family of problems is then said to be *successful with non-negligible probability* if the attack succeeds with a probability that is greater than $1/p(k)$, for all sufficiently large values of k, where p is a polynomial [3].

The rôle played by probability in our framework is rather different: We deliberately employ probabilistic techniques to replace the need for complex logical structure (and, indeed, expressions) depending upon detailed systems knowledge. Thus, probability for us serves to simplify the applied models that we are interested in. We can do this because we are not so concerned with the detailed specification of system components and their correctness — our concern lies more with the large-scale requirements specification of services, their operational delivery, and economic value. This philosophy has been deployed to great effect in modelling with the **Demos 2000** system.

2 A Mathematical Framework

The notion of *resource* is a basic one in many fields, including economics, engineering, and the humanities, but it is perhaps rather clearly illuminated in the computing sciences. The location, ownership, access to, and consumption of resources are central concerns in the design of systems, such as networks, within which processors must access devices such as file servers, disks, and printers, and in the design of programs, which access memory and manipulate data structures, such as pointers.

In recent years, it has been demonstrated that a simple, semantic model of the notion of *resource*, due to Pym and O'Hearn [36, 38, 39], can be a highly effective tool for analyzing the meaning of computations that require the controlled sharing of data. The leading examples are perhaps Reynolds' separation logic [40], and O'Hearn's analysis of Idealized Algol and Syntactic Control of Interference [35] and recent work on concurrent separation logic [34]. More recently, Collinson, Pym, and Robinson [14, 13] have shown how resource semantics can be used to explain ML-like languages with multiplicative quantifiers, giving rise to a form of polymorphism that can be used to capture cleanly a range of desirable region- and location-based language features.

From the perspective of process theory, Cardelli, Gordon, and others [12, 11, 10], locations and, indeed, resources, are represented by certain classes of process terms. Our approach is quite different: we prefer to represent processes, resources, and locations in terms that are directly motivated by those of their properties that we wish to capture, leading to both conceptual and computational simplifications.

2.1 Resource Semantics and Logic

A mathematical account of a useful notion of resource can be given using logic. Our starting position is that the following properties are reasonable requirements for a simple model of resource [38, 39, 16]:

- A set $\mathbf{R}$ of resource elements;
- A (partial) combination, $\circ : \mathbf{R} \times \mathbf{R} \rightharpoonup \mathbf{R}$ of resource elements;
- A comparison, $\sqsubseteq$, of resource elements; and

– A zero resource element, e.

In the usual spirit and methodology of mathematically modelling, these conceptually evidently well-motivated properties correspond well to a wide a range of natural examples [39, 38]. Mathematically, we obtain this structure as a pre-ordered partial commutative monoid,

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

subject to the condition that if $r \sqsubseteq s$ and $r' \sqsubseteq s'$, then $r \circ r' \sqsubseteq s \circ s'$, and, recalling the preordering of a Kripke structure [23, 24], call it a *Kripke resource monoid*, or KRM, with worlds being resources. The ordering $\sqsubseteq$ gives rise to an equality.

A simple example is provided by the natural numbers, here including 0,

$$\mathcal{N} = (\mathbb{N}\,,\,+\,,\,0\,,\,\leq),$$

in which combination is given by addition, with unit 0, and comparison is given by less than or equals. This is an example of resource as *cost*.

Of quite direct relevance to our concerns is the 'basic separation model' [39, 40]. Suppose we are given an infinite set $Res = \{r_0, r_1, \ldots\}$. We think of the elements of Res as primitive resources, or resource IDs, that can be allocated and deallocated. The partial monoid structure is given by taking a world to be a finite subset of Res, and $\circ$ to be union of disjoint sets. In more detail, where $\uparrow$ denotes undefinedness (and $\downarrow$ definedndess),

$$m \circ n = \begin{cases} m \cup n & \text{if } m \cap n = \emptyset \\ \uparrow & \text{otherwise.} \end{cases}$$

The unit of $\circ$ is $\{e\}$, and we take $\sqsubseteq$ to be equality. This example is the basis of Ishtiaq and O'Hearn's pointer logic [21] and Reynolds' separation logic [40].

The composition and ordering structure lifts to sets of resource elements. Let $\wp(\mathbf{R})$ denote the powerset of $\mathbf{R}$ and let $R, S \in \wp(\mathbf{R})$. Then define, for example,

$$R \circ S \;=\; \begin{cases} \{\, r \circ s \;\mid\; r \in R \text{ and } s \in S \,\} & \text{if each } r \circ s \downarrow \\ \uparrow & \text{otherwise,} \end{cases}$$

with unit $\{e\}$ and, for example, $R \sqsubseteq S$ iff, for all $r \in R$, there is $s \in S$ such that $r \sqsubseteq s$. Such sets of resources are a convenient level of abstraction for our present purposes, for which we shall require no further special properties. We might also require that $R \circ S$ be defined only if R and S are disjoint. We write R_1, R_2 for the union of R_1 and R_2, and emphasize that composition is quite different from union. Our notational choices should be clear *in situ*. Other constructions, based on Kripke resource monoids, might also provide a basis for a calculus and logic. The space of choices is, however, quite large, so that a discussion of it is beyond our present scope. More generally, we might take a more complex structure of resources [37].

Kripke resource monoids provide the basis for the semantics of **BI**, the logic of bunched implications [36, 38]. The judgement $r \models \phi$, for $r \in \mathbf{R}$, is read as 'resource element r is sufficient to support proposition ϕ'. The ordering structure admits the usual Kripke semantics for the usual, additive, connectives ($\top$, $\wedge$, $\bot$, $\vee$, $\rightarrow$) of intuitionistic

logic and, in the discrete case, classical logic.The monoidal structure admits a semantics for a multiplicative conjunction, $*$, given by

$$r \models \phi_1 * \phi_2 \text{ iff there are } s_1 \text{ and } s_2 \text{ such that } s_1 \circ s_2 \sqsubseteq r, \text{ and } s_1 \models \phi_1 \text{ and } s_2 \models \phi_2.$$

The semantics of the multiplicative conjunction, $*$, is interpreted as follows: the resource r is sufficient to support $\phi_1 * \phi_2$ just in case it can be divided into resources s_1 and s_2 such that s_1 is sufficient to support ϕ_1 and s_2 is sufficient to support ϕ_2. The assertions ϕ_1 and ϕ_2 — think of them as expressing properties of programs — *do not share* resources. In contrast, in the semantics of the additive conjunction, $r \models \phi_1 \wedge \phi_2$ iff $r \models \phi_1$ and $r \models \phi_2$, the assertions ϕ_1 and ϕ_2 may *share* the resource m. Along with the multiplicative conjunction comes a multiplicative implication, $-\!\!*$, given by

$$r \models \phi -\!\!* \psi \text{ iff for all } s \text{ such that } s \models \phi, r \circ s \models \psi.$$

The semantics of the multiplicative implication, $-\!\!*$, may be interpreted as follows: the resource r is sufficient to support $\phi -\!\!* \psi$ just in case for any resource s which is sufficient to support ϕ the combination $r \circ s$ is sufficient to support ψ. We can think of the proposition $\phi -\!\!* \psi$ as (the type of) a function and the proposition ϕ as (the type of) its argument. The resources then describe the cost of applying the function to its argument in order to obtain the result. The function and its argument *do not share* resources.

In contrast, in the semantics of additive implication, the function and its argument may *share* the resource s. Intuitionistically, $r \models \phi \rightarrow \psi$ iff for all $r \sqsubseteq s$, $s \models \phi$ implies $s \models \psi$; classically, $r \models \phi \rightarrow \psi$ iff $r \models \phi$ implies $r \models \psi$.

In Figure 1, we illustrate how separation logic can be used to allocate resources amongst a collection of users. In this example, there is a single, shared computational resource (system S) in which user X_i has, perhaps dynamically in the context of utility computing, resource R_i. (Naturally, resources can be all manner of things; the requirements of composing and comparing can typically be met very easily in a wide range of circumstances, often trivially.)

Here we intend that the systems $X_1, X_2, \ldots, X_m$ will be executing processes accessing the central resources. If the systems X_i are required to satisfy properties ϕ_i, such as availability or security assertions, then the whole system, in respect of these properties, is described by

$$(R_1, S) \circ (R_2, S) \circ \ldots \circ (R_m, S) \models \phi_1 * \phi_2 * \ldots * \phi_m.$$

To see how this works, consider that unpacking the formula $\phi_1 * \phi_2 * \ldots * \phi_m$ into its component parts involves dividing the resources m ways, corresponding to the m components of 'total' resource, each of which contains a reference to the shared resource, S. We shall return to this example later when we discuss access control.

2.2 A Calculus of Resource Processes: SCRP

Within a process algebra [28, 4, 20, 29], the common representation of resource is as a separated process. For instance, a semaphore is represented as a two-state process, representing whether the token is currently available or not. There have been extensions

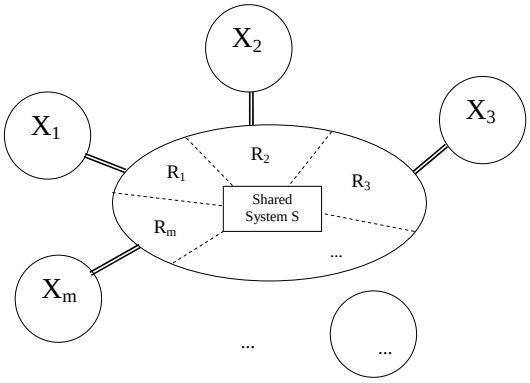

Fig. 1. Splitting resources amongst a number of distributed, concurrent systems

[9] that attempt to model resource explicitly but these approaches carry both the communication structures of the process algebras alongside the representation of resource. We take the view that resource *is* the fundamental organizing principle of the underlying calculus, an approach taken within process-oriented discrete event languages [5, 6]. There has been a demonstration that Milner's calculus SCCS can support a compositional view of resource directly [43]. It is clear, however, that this approach still contains all of the fundamental action structures of SCCS. Our approach is to consider the co-evolution of resources, as discussed above, and processes, in the sense of SCCS, with synchronization being constrained by the availability of resources and with resources being modified by the occurrence of actions. Our starting point for our calculus of resource process is Milner's synchronous calculus of communicating systems, SCCS [28]. Note that the asynchronous calculus CCS is a *sub*-calculus of SCCS.

We give an informal description of the Synchronous Calculus of Resources Processes, or **SCRP**, introduced by Pym and Tofts [37]. The language of process element of **SCRP** follows the notation of SCCS and the π-calculus:

- **1**, the unit process;
- $a : E$ a process that performs the action a to become the process E;
- $E + F$ a process that evolves as E or as F, unit **0**;
- $E \times F$ a process that synchronously uses resources as E and F;
- $C \stackrel{\text{def}}{=} E$ is the definition of a constant C, allowing recursive processes to be defined;
- $\nu(S)E$ a process with a local, or hidden, evolution relative to resource S and enabling and modification functions ρ and μ, explained below.

The main development in **SCRP** is to view the statement $E \stackrel{a}{\longrightarrow} E'$ as meaning that by using resource required for the action a to be enabled, the process E evolves to E', with a corresponding modification of the available resource. We implement this change of perspective by supposing the existence of an *enabling* function ρ which assigns to each action a the resources $\rho(a)$ required for it to be enabled and a *modification* function

μ which assigns to each action a and each collection R of resources the collection of resources $\mu(a, R)$ which results from performing a with resource R. Thus **SCRP**'s operational rule for action prefix essentially takes the form

$$\frac{}{R\,,\, a : E \xrightarrow{a} \mu(a, R)\,,\, E} \quad R \text{ is at least } \rho(a).$$

Synchronization is achieved by the requiring that a parallel composition of actions, $a\#b$, be possible only if the resource environment can be decomposed to support a and b separately. Thus **SCRP**'s operational rule for parallel composition essentially takes the form

$$\frac{R_1\,,\, E_1 \xrightarrow{a_1} \mu(a_1, R_1)\,,\, E_1' \qquad R_2\,,\, E_2 \xrightarrow{a_2} \mu(a_2, R_2)\,,\, E_2'}{R\,,\, E_1 \times E_2 \xrightarrow{a_1\#a_2} \mu(a_1\#a_2, R)\,,\, E_1' \times E_2'} \quad R = R_1 \circ R_2 \text{ is defined;}$$

that is, it must be possible to decompose R into the resources R_1 and R_2, the resources required to support a_1 and a_2 simultaneously, though we admit the possibility of an equality between R_1 and R_2, so allowing sharing as required. Note that synchronization is regulated by resources, in constrast to ACSR [9], in which instantaneous events provide the basic synchronization mechanism. Note also that, in contrast to **SCRP**'s local conditions, Gastin and Mislove [18] require a global construction for synchronization.

One fundamental consequence of this approach is that we should wish to maintain all of the interactions that lead to the current resource use transition within a process. In some sense, we need to know how the current resource utilization can be decomposed. So we must abandon the elegant use of the free abelian group of actions within SCCS to describe actions, restricting **SCRP** to the more basic free abelian monoid [28],

$$\mathcal{A} = (Act\,,\, \#\,,\, \mathbf{1}).$$

If we were to take an abelian group, then an action a might result from the composition $a\#b^{-1}$ and b thus, in some sense, making use of more resource. Taking resource as the basic organizing principle, this form of hiding makes decomposition difficult to track. Nevertheless, **SCRP**'s formulation permits the formulation of compound atomic actions (see definition of ρ, below) which are able to emulate the difficult wait-until aspect of discrete event systems modelling languages such as **Demos 2000**.

SCCS, in common with CCS, uses a notion of *restriction*. In **SCRP**, a more natural concept is that of a *local* action, in which a collection of resources is available only to the process to which it is bound. Informally, the operational rule should take the form

$$\frac{R \circ S\,,\, E \xrightarrow{a} R' \circ S'\,,\, E'}{R\,,\, \nu(S)E \xrightarrow{(\nu S)a} R'\,,\, \nu(S')E'},$$

where '$\nu(S)a$' denotes the action a without the components of it that are associated with the bound resource S. These components are 'hidden' in the subsequent evolution.

To make all this work mathematically, we need the following set-up for enabling and modification:

– A family of partial functions $\rho : Act \rightharpoonup \wp(\mathbf{R})$ that assign to each action, a, a set of resources. Think of this as the set of resources required in order for a to be enabled. We require some mathematical properties to ensure that these functions are well-behaved [37];
– A family of partial functions $\mu : Act \times \wp(\mathbf{R}) \rightharpoonup \wp(\mathbf{R})$, which should be understood as describing the modification to a set R of resources caused by the execution of the action a. Again, we require some mathematical properties to ensure that these functions are well-behaved [37].

Notice the *separation conditions* in the $Prod$ and $Hide$ rules. In the $Prod$ rule, we ensure that the composite resource is defined. The non-interference of the components of the composition can be enforced by requiring also that $R \circ S$ be defined only if R and S are disjoint (cf. separation logic [40]). In the $Hide$ rule, in which we implicitly intend that the enabling function ρ accounts for the non-hidden actions and σ for the hidden actions, such a realization of the definedness condition would ensure that the bound resources be not accessible by processes in the environment.

The rules for non-determinism and for constants, with which we can form recursive definitions, seem quite familiar and are quite straightforward:

$$\frac{R\,,\, E_i \xrightarrow{a} \mu(a,R), E_i'}{R\,,\, E_1 + E_2 \xrightarrow{a} \mu(a,R), E_i'}\; i = 1,2 \qquad Con \quad \frac{R\,,\, E \xrightarrow{a} \mu(a,R)\,,\, E'}{R\,,\, C \xrightarrow{a} \mu(a,R)\,,\, E'}\; C \stackrel{\text{def}}{=} E.$$

Bisimulation for **SCRP**, $R\,,\, E \sim R\,,\, F$, is defined in the usual way. Note that, for now, we consider the processes E and F relative to the same resource environment. As usual, we suppress the enabling and modification functions.

Definition 1 ([37]). *Bisimulation, $\sim$, is the largest binary relation on resource–process pairs, R, E such that if $R, E \sim R, F$, then (i) $R, E \xrightarrow{a} \mu(a,R), E'$ implies, for some F', $R, F \xrightarrow{a} \mu(a,R), F'$ and $\mu(a,R), E' \sim \mu(a,R), F'$; and (ii) $R, F \xrightarrow{a} \mu(a,R), F'$ implies, for some E', $R, E \xrightarrow{a} \mu(a,R), E'$ and $\mu(a,R), E' \sim \mu(a,R), F'$.*

Theorem 1 ([37]). *Bisimulation of resource processes is a congruence.*

That is, in our setting, that if $R\,,\, E \sim R\,,\, F$, then, for all evident terms a, G, and S, $R\,,\, a : E \sim R\,,\, a : F$, $R\,,\, E + G \sim R\,,\, F + G$, $R\,,\, E \times G \sim R\,,\, F \times G$, and $R\,,\, \nu(S)E \sim R\,,\, \nu(S)F$.

We will not discuss here the equations satisfied by **SCRP**'s processes. Throughout our presentation we have an intention that the calculus will be used to represent *implementation* and that an extended resource logic will be used to represent *requirements*. As a consequence, there is little need to reason directly within the process calculus exploiting an equational theory. The familiar equations, such as commutativity, associativity, etc., do indeed hold. But we cannot expect to obtain an expansion theorem — relating concurrency and non-determinism — as an equivalence. The main reason for this is that when we consider the constituent parts of a parallel composition we will have a particular allocation of resources to each of those parts. When we form the parallel composition we naturally form a (typically larger) compound resource, it is clear that this could have been divided in many ways other than that which we chose to do the

original proofs of the behaviours of the sub-components. So in **SCRP**-based settings, we obtain expansion only in the setting of an inequational theory, in which one works not with bisimulation but with simulation.

Finally, we remark that **SCRP** provides an appropriate mathematical framework in which to give a semantics to **Demos 2000** in the sense of that given in SCCS [28, 7, 8], in which the stochastic data capture is, with little loss of generality, elided. Systems such as **Demos 2000**, however, implement a process-theoretic view of the world. Not only is the dynamics of systems represented as processes but so too are the essentially static resource components. We would argue that this situation is conceptually unsatisfactory. Moreover, pragmatically, the computational cost of modelling interactive systems is, typically, dominated by the handling of the resource components.

2.3 A Modal Logic of Resource Processes, MBI

Process calculi such as SCCS and CCS come along with a modal logic, usually called Hennessy-Milner logic, with a semantic judgement of the form $E \models \phi$, read as 'process E has property ϕ'. The language of propositions typically consists of classical conjunction, disjunction, and negation, together with modalities $\langle a \rangle$ and $[a]$ for describing the properties of evolutions $E \xrightarrow{a} E'$.

In our setting, with an explicit model of resources and a corresponding logic, we are able to work with a judgement $R\,,\, E \models \phi$, read as 'relative to the available resources R, process E has property ϕ'.

In this setting, we can immediately recover the familiar classical connectives:

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

$$R\,,\, E \models \neg\phi \;\text{ iff }\; R\,,\, E \not\models \phi.$$

The corresponding intuitionistic connectives are also available:

$$R\,,\, E \models \phi \supset \psi \;\text{ iff }\; \text{for all } R \sqsubseteq S \text{ and all } E \sim F,\, S\,,\, F \models \phi \text{ implies } S\,,\, F \models \psi.$$

The intuitionistic version of the universal quantifier is, of course, obtained similarly. Clearly, some variations are possible here.

Hennessy-Milner logic's necessitation modality, $[a]$ is also recovered quite simply:

$$R\,,\, E \models [a]\phi \;\text{ iff }\; \text{for all } R\,,\, E \xrightarrow{a} \mu(a, R)\,,\, E', \text{ s.t. } \rho(a) \sqsubseteq R,\, \mu(a, R)\,,\, E' \models \phi.$$

The possibility modality, $\langle a \rangle$, is recovered similarly. Note, however, that the resource element is important in this definition: the action a must be enabled by the available resource.

In our richer logical setting, we are able to obtain a finer analysis of this judgement than is available in Hennessy-Milner logic. Specifically, we obtain, essentially, the following characterization of parallel composition, denoted by $\times$, as in SCCS:

$$\begin{aligned} R\,,\, E \models \phi_1 * \phi_2 \text{ iff there are } R_1 \text{ and } R_2 \text{ such that } R_1 \circ R_2 = R \\ \text{and there are } E_1 \text{ and } E_2 \text{ such that } E_1 \times E_2 \sim E, \\ \text{such that } R_1\,,\, E_1 \models \phi_1 \text{ and } R_2\,,\, E_2 \models \phi_2. \end{aligned}$$

As well as these propositional connectives, we also get multiplicative modalities. The necessitation is given by

$$R\,,\,E \models [a]_\nu \phi \text{ iff for all } R \circ S\,,\,E \xrightarrow{a} \mu(a, R \circ S)\,,\,E', \text{ s.t. } \rho(a) \sqsubseteq R \circ S \text{ and } R \circ S\,,\,E' \models \phi$$

and should be understood as characterizing the additional resource required for ϕ to hold if it is guarded by the action a. Again, there are clearly some choices here.

Finally, by working with **BI**'s multiplicative quantifiers, we are also able to characterize the notion of local resource:

$$R\,,\,E \models \forall_\nu x.\phi \text{ iff for all } S\,,\,F \text{ s.t. } R\,,\,E \sim R\,,\,\nu(S)F,\, R \circ S\,,\,F \models \phi[b/x],$$

for a suitable (quite straightforward) definition of the term b (see [37]), with a similar clause for $\exists_\nu$. That is, the hiding construction in **SCRP**, $\nu(S)E$, that binds the resource S to E is characterized by the multiplicative quantifiers: the quantified formula specifies that the process must have a certain quantity of private resource.

The logical characterization of bisimulation provided by Hennessy-Milner logic for a process calculus such as CCS [29] takes the form

$$E \sim F \;\text{ iff }\; \text{for all } \phi,\, E \models \phi \text{ iff } F \models \phi.$$

Such a theorem is available for the finer analysis of process equivalence and logical equivalence provided by **SCRP** and **MBI**. More specifically, our result, expressed as Theorems 2 and 3, shows that **MBI** provides explicit characterizations of the concurrent and local structure of a system, via the definitions of $\models$ for the connective $*$ and the multiplicative quantifiers, $\forall_\nu$ and $\exists_\nu$, respectively.

Definition 2 ([37]). *Let Γ be a set of* **MBI** *formulæ. The equivalence $\equiv_\Gamma$ between* **SCRP** *processes is defined by $R\,,\,E \equiv_\Gamma R\,,\,F$ iff $\{\phi \in \Gamma \mid R\,,\,E \models \phi\} = \{\psi \in \Gamma \mid R\,,\,F \models \psi\}$.*

We have the evident definition: $R\,,\,E \equiv_{\mathbf{MBI}} R\,,\,F$ iff for all Γ, $R\,,\,E \equiv_\Gamma R\,,\,F$.

Theorem 2 ([37]). *If, for all R, $R\,,\,E \sim R\,,\,F$, then, for all R, $R\,,\,E \equiv_{\mathbf{MBI}} R\,,\,F$.*

Theorem 3 ([37]). *If, for all R, $R\,,\,E$ and $R\,,\,F$ are image-finite and if, for all R, $R\,,\,E \equiv_{\mathbf{MBI}} R\,,\,F$, then, for all R, $R\,,\,E \sim R\,,\,F$.*

Unfortunately, although the first-order quantifiers are naturally present in our setting, it seems that they are insufficient to capture the non-image-finite case. Just as for Hennessy-Milner logic, it seems that to handle non-image finite resource processes, we must either use an infinitary propositional logic or introduce fixed points, as in the modal μ-calculus [41].

A similar objective is encountered in the work of Cardelli and Caires [11] in which a 'spatial logic', in many ways similar to **MBI** but lacking an explicit notion of resource, is used to model the asynchronous π-calculus. A detailed exploration of possible relationships between this work and ours — perhaps via particular choices of resource monoid — is beyond our present scope.

Another approach to resources, in a synchronous setting, is that of Brémond-Grégoire and Lee's ACSR [9]. Our approach is more foundational, starting from a logically well-founded model of resource and developing a theory, in the modelling context described above, of the interaction between processes and resources. A similar point of view may be found in the work of Gastin and Mislove [18].

3 Systems Integrity and Access Control

Taking our lead from [2], distributed ICT systems security depends upon the following three layers and their principal concerns:

1. **Trusted computing**: Known systems with defined functional capabilities;
2. **Authentication**: Known identity of people with defined roles;
3. **Authorization**: Known roles and functions of people and systems using resources within a defined organizational context.

Because of these dependencies, the provision of authorization requires (some form of) authentication and, in turn, the provision of authentication depends upon (some form of) trusted computing. Thus, access control is mainly concerned with authorization. Crucially, effective access control requires not only knowledge of what people and systems do with resources, but also the intended (business) goals of what is done using those resources — such matters influence the access-control policy and the decisions made.

Stated this way, distributed authorization in a business context involves understanding business functions, what resources can be used for and the process connections that entails. There are some semi-formal techniques, such as Domain Based Security (DBSy)[26, 27], that help provide ways of capturing and assessing the network security requirements upon communications and more generally, network services, within large distributed organizations such as government departments and large corporations. The approach focuses upon how the *business* itself requires information and it's processing to be compartmentalized — that is, *network separations*, *services aggregation* and *compartmentalization-in-the-large*. The framework we report here is thus a tentative contribution towards a more formal account of these issues.

The remainder of this section introduces a somewhat rudimentary example, allowing us to bring concepts from resource semantics to bear upon access control, and to also motivate why some abstraction of location is necessary.

3.1 Our Basic Example

The basic scenario is concerned with a customer wishing to access some protected data on a corporate database server over a network. This is illustrated in Figure 2.

This diagram attempts to document various relationships graphically; for example, the hexagons indicate classes/roles of people who can interact and have some responsibilities for systems and services (indicated by circles). The double dashed lines indicate this association between people and systems. Finally, the double-headed arrows indicate message- or data-flows between systems. We note in passing how the evident complexity of diagrams like this, even for apparently simple examples such as this one, amply illustrates the need for a more formal approach to such questions.

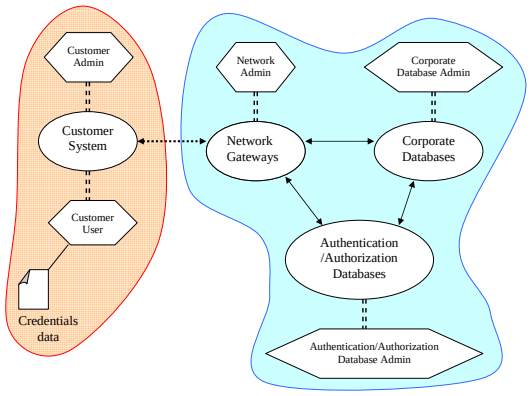

Fig. 2. Basic set-up

To access material from the corporate database, customers must be able to prove their (remote) identity and that it matches to some appropriate customer identity. This is done by customers sending appropriate credentials upon request.

A typical interaction between a customer and the corporate database uses a network gateway that checks credentials for all sessions using the network. If the credentials are acceptable, customers may proceed to access the corporate database. However, certain queries to the database are privileged and require various clearances - these are also encapsulated into the credentials.

Thus, to check that the credentials in each case are appropriate, there is also an authentication/authorization database (AADB) that must be actively checked online.

We have used **Demos 2000** to provide a basic probabilistic model of the system above (see Appendix B). Probability is used here to *abstract away* from a more detailed treatment of security-related features such as user accounts, personal profiles and individual options that would be typically maintained within the AADB.

Although certain security details are not concretely represented, the **Demos 2000** model still retains the most significant security-related feature — the critical dependency of both network gateway and database access upon the *availability* of the AADB. This tells the analyst that if the AADB were to fail, then the entire system would also fail. In other words, our **Demos 2000** model demonstrates critical dependencies, even at this quite high level of abstraction.

Additionally, because the model can also be simulated, we could explore the effect of other trade-offs such as the effect of providing some form of secure database replication of the AADB to improve resilience etc. Naturally, such a strategy would have impact upon capital infrastructure costs as well as operating costs — and these could also be captured within **Demos 2000**. As it stands, this model is useful for exploring security-related trade-offs around availability and resilience of the support system. It is also clear that at some later stage we may become interested in modelling some of the details we used probability to abstract away from.

3.2 Our Example, Further Refined

We can further refine the above example by indicating *scope of control*, an abstract form of location, as illustrated in Figure 3. For example, we may wish to associate each of the core capabilities — gateways, databases, authentication — with their administrative support. The point being that systems administration should be localised to each function, but at the same time implementing global policy requirements. Such managerial coordination will involve communications between the administrators and with the overseeing corporate systems management.

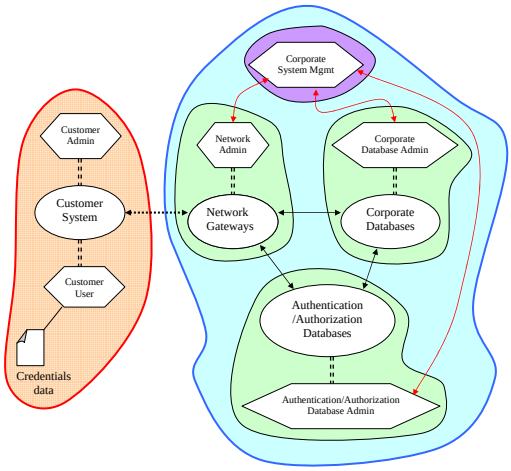

Fig. 3. Refined Security Example

This refinement suggests a need for a notion of *location* that allows the system modeller to capture his chosen level of detail and chosen connectivities.

4 Location in the Mathematical Framework

The literature on the theory of computation in general, and on concurrent computation in particular, contains a wide range of approaches to the notion of location, and the range of technical complexity required varies greatly. In the work, described above, related to separation logic and bunched polymorphism, for example, the starting point is simply a *set* of (names of) locations. In both the works of Cardelli and Gordon [12], on ambients, and of Jensen and Milner [22], on bigraphs, the notion of location (and, indeed, of resource) is captured within a behavioural framework involving all the complexity of, for example, the π-calculus [30]. Such an approach represents, perhaps, a quest for a grand unified theory of computational structures. Just as with our motivation for separating resources from processes, our ambitions are more prosaic: we seek

a conceptually direct, technology for capturing the various features of systems that are relevant to addressing system-scale questions of performance, integrity, and cost (and so of economic viablity).

Whilst admitting that some modelling tasks might require rather more complex notions of location, we begin here by suggesting a basic framework, in the context of our existing analysis of resources and processes and our modelling philosophy, that provides the essential features needed to begin an analysis.

Recall that our resource process judgements are of the form $R\,,\ E \xrightarrow{a} R'\,,\ E'$, for the operational semantics of **SCRP**, and $R\,,\ E \models \phi$, for the logic **MBI**. We enrich these judgements to have the form

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

read as, 'with resources R at starting location L, the process E evolves to E', resulting in resources R' at finishing location L''. Note that we require a connectivity property between L and L', and that the judgement describes just a local evolution. For this conception to be sensible, it seems, following the same modelling philosophy used to derive our assumptions about resources, we need

- a notion of *sublocation*, $L \preceq M$,
- *substitution* of locations, $M[L'/L]$, of location L' for a sublocation L of M,
- a notion of *connection* between locations, and
- a *product* of locations.

Sublocations arise from, among other things, the need, typically, for a local evolution to describe what happens to the starting location as a result of the evolution. A substitution is required to ensure that we capture an appropriate compositionality of systems. A product is needed to capture how concurrent actions may draw upon resources from distinct locations. This idea of location captures both the physical and the virtual.

One simple way to realize these requirements is to take locations to be finite, (directed) graphs. Sublocations arise a subgraphs, substitution is given by replacement of a subgraph by a graph of matching arity — that is, matching (directed) arcs — and product is given by a suitable choice of graph product (there are many, including a categorical product and a range of monoidal products). Two sublocations L and M of a location N are connected — taking due account of directedness as necessary — if there is an arc linking a vertex of L to a vertex of M. We believe that the constructs of Cardelli and Gordon [12], Jensen and Milner [22], and Galmiche and Méry [17] can be considered to satisfy these requirements.

Returning to our development from resource-processes to location-resource-processes, it is clear that we must adapt the formulations of the enabling and modification functions. Recalling the basic form of the axiom case of SCRP's operational semantics, we can see that we require, with ρ and μ having the evident types,

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

with the following definitions: $\rho(a, L) \sqsubseteq R$, $\mu(a, L, R) = (L', R')$.

Note that this framework permits resources to be associated with a location that is either a single vertex or a whole graph, reflecting the choice of degree of abstraction.

We will not reconstruct all of **SCRP** and **MBI** in the presence of locations. Rather, we will illustrate a few interesting points. The most obvious questions arise around the interaction of location and hiding and location and concurrent composition.

4.1 Resource Distribution and Allocation

In **SCRP**, one possible formulation allows the concurrent compostion of resource-processes at a common location, that is

$$\frac{L\,,\,R\,,\,E \xrightarrow{a} L\,,\,R'\,,\,E' \quad L\,,\,S\,,\,F \xrightarrow{a} L\,,\,S'\,,\,F'}{L\,,\,R \circ S\,,\,E \times F \xrightarrow{a\#b} L\,,\,R' \circ S'\,,\,E' \times F'},$$

where $R' = \mu(a, R)$, etc., and under appropriate definedness/separation conditions. Another choice is to exploit the availability of a product, $\bowtie$, of locations and allow, subject to appropriate conditions,

$$\frac{L\,,\,R\,,\,E \xrightarrow{a} L\,,\,R'\,,\,E' \quad M\,,\,S\,,\,F \xrightarrow{a} L\,,\,S'\,,\,F'}{L \bowtie M\,,\,R \circ S\,,\,E \times F \xrightarrow{a\#b} L \bowtie M\,,\,R' \circ S'\,,\,E' \times F'}.$$

Finally, one might generalize each of these to permit the evolution of locations — L to L', M to M'. We conjecture that these choices are all that are required.

In **SCRP**, hiding binds resources locally to a process: In $S\,,\,\nu(R).E$, the ambient, shared system resources are S and E has, additionally, private access to R. This can be seen pictorially by reconsidering our first pictorial example, as depicted in Figure 4.

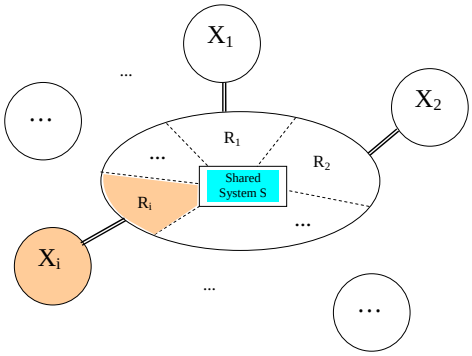

Fig. 4. Local Resources

Here we envisage the system X_i running some process E_i. It has access to shared resources S and local resources R, and is described as $S\,,\,\nu(R_i).E_i$. Another system X_j might be described as $S\,,\,\nu(R_j).E_j$, with R_i and R_j satisfying a separation condition.

4.2 Access Control Revisited

Given this structural set-up, how do we ask whether our (model of our) system supports our access control policy? To see this, and as an example, we consider descriptions of access control policies such as in Binder [15], as described by Abadi [1]. Consider the system described above and illustrated in Figure 4. Each of the systems X_i may have an access control policy, expressed as logical formulæ σ_{X_i}, as surveyed by Abadi [1]. Such formulæ amount to certain logical combinations of predicates such as `may-access(p,o,r)`, which would hold whenever the policy gives principal `p` the right `r` on object `o`. Then, if the whole system — expressed as $L\,,\,R\,,\,E$, where E is *essentially* the concurrent composition of the m E_is — is to support all of the required access control policies, we must have

$$L\,,\,R\,,\,E \models \sigma_{X_1} * \sigma_{X_2} * \ldots * \sigma_{X_m}\,.$$

This will hold provided the system $L\,,\,R\,,\,E$ can be decomposed in such a way as to support all of the policy requirements separately; that is, each

$$L_i\,,\,R_i\,,\,E_i \models \sigma_{X_i}$$

holds, for some well-defined decomposition.

Using tableaux systems for **MBI**, similar to those available for **BI** [16], we aim to do efficient model checking of access control policies exploiting system models incorporating location — see [33] for an example.

5 Some Technical Directions

We have presented a very high-level overview of a wide-ranging project in modelling techniques for systems integrity. There are many research directions that are being explored in much greater detail:

- The mathematical theory of **SCRP** and **MBI** [37], with and without locations;
- Tools, in the style of **Demos 2000** and of model checking to support modelling;
- Constructs that naturally handle ideas such as *rôles* and *impersonation* in access control, building on ideas discussed by Abadi et al. [2, 2]. For example, the idea of principal E in rôle F, or 'E quoting F', can be made precise as a form of non-commutative concurrent composition, $E \propto F$, in our setting:

$$\frac{R\,,\,F \xrightarrow{a} R'\,,\,F' \quad S\,,\,E \xrightarrow{a} S'\,,\,E'}{S\,,\,E \propto F \xrightarrow{a} S'\,,\,E' \propto F'}\; R \sqsubseteq S, \quad S\,,\,E \sim S\,,\,F,$$

  Interestingly, the non-commuativity arises rather naturally via our explicit representation of resources, not present in [2]. Note that the bisimulation could be relaxed to simulation, a choice not readily available in Abadi et al.'s calculus of principals. Building on this operational construct, we are able to recover the idea of 'principal E *says* ϕ' as a form of modality in **MBI**, $\{E\}\phi$, associated directly with $\propto$:

$$R\,,\,G \models \{E\}\phi \text{ iff for some } F \text{ s.t. } R, G \sim (R, E \propto F),\, R\,,\,F \models \phi.$$

That is, E says ϕ holds for G just in case G is of the form E quoting F and F supports ϕ (all relative to resources R). We can enrich this analysis with our notion of location. Abadi et al. proceed to analyze a range of derived constructions, involving ideas such as delegation and certificates. These ideas remain to be explored.

The framework we have sketched thus allows us to begin to analyze the concept of *stewardship*; that is, the idea that when customers entrust their resources, such as business-critical corporate data, to, for example, a utility computing service, they expect their data to be cared for appropriately (confidentially, with integrity, and with high availability) when it is processed by the utility providers' resources. Thus we need an account of the interactions between these two classes of resource within the utility processing environment.

Acknowledgements. We are grateful to Martin Sadler, Richard Taylor, and Mike Yearworth for discussions that have provided the context for this work. We are grateful to Matthew Collinson, Peter O'Hearn, and Chris Tofts for relevant technical discussions.

References

1. Martín Abadi. Logic in access control. In *Proc. LICS 2003*, 228–233, IEEE, 2003.
2. Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Prog. Lang. Sys.* 15(4):706–734, 1993.
3. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.
4. J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In *Proc 11th ICALP, LNCS 172*, 1984.
5. G. Birtwistle. *Demos — discrete event modelling on Simula.* Macmillan, 1979.
6. G. Birtwistle. Demos implementation guide and reference manual. Technical Report 81/70/22, University of Calgary, 1981.
7. G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part I πDemos. *Trans. Soc. Comp. Sim.* 10(4):299–333, 1993.
8. G. Birtwistle and C. Tofts. A denotational semantics for a process-based simulation language. *ACM ToMaCS*, 8(3):281 – 305, 1998.
9. Patrice Brémond-Grégoire and Insup Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoret. Comp. Sci.* 189(1–2):179–219, 1997.
10. Luís Caires and Luca Cardelli. A spatial logic for concurrency-ii. *Theoret. Comp. Sci.* 322(3):517–565, 2004.
11. L. Cardelli and L. Caires. A spatial logic of concurrency (part i). *Information and Computation*, 186(2):194–235, 2003.
12. L. Cardelli and A. Gordon. Anytime, anywhere: modal logics for mobile processes. In *Proc. 27th POPL*, 2000, ACM.
13. M. Collinson and D. Pym. A bunched approach to the semantics of regions and locations. In *Proc. SPACE 2006, Charleston, South Carolina*, 2006.
14. M. Collinson, D. Pym, E. Robinson. On bunched polymorphism. LNCS 3634: 36-50, 2005.
15. John DeTreville. Binder, a logic-based security language. In *Proc. 2002 IEEE Symposium on Security and Privacy*, pages 105–113, 2003.
16. D. Galmiche, D. Méry and D. Pym. The semantics of **bi** and resource tableaux. *Math. Struct. Comp. Sci.*, 15:1033–1088, 2005.
17. D. Galmiche and D. Méry. Resource Graphs and Countermodels in Resource Logics. *Electronic Notes in Computer Science* 125, 2005.

18. P. Gastin and M. Mislove. A simple process algebra based on atomic actions with resources. *Mathematical Structures in Computer Science*, 14:1–55, 2004.
19. L.A. Gordon and M.P. Loeb. *Managing Cybersecurity Resources: A Cost-Benefit Analysis*. McGraw Hill, 2006.
20. C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
21. S.S. Ishtiaq and P. O'Hearn. **BI** as an assertion language for mutable data structures. In *28th ACM-SIGPLAN Symposium on Principles of Programming Languages, London*, pages 14–26. Association for Computing Machinery, 2001.
22. O.H. Jensen and R. Milner. Bigraphs and mobile processes (revised). Technical report, University of Cambridge, 2004. UCAM-CL-TR-580, ISSN 1476-2986.
23. S. A. Kripke. Semantical considerations on modal logic. *Acta Phil. Fenn.*, 16:83–94, 1963.
24. S. A. Kripke. Semantical analysis of intuitionistic logic I. In J. N. Crossley and M. A. E. Dummett, editors, *Formal Systems and Recursive Functions*, 92–130. North-Holland, 1965.
25. Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Trans. on Comp. Sys.*, 10(4):265–310, 1992.
26. C. L.Robinson. Security requirements models to support the accreditation process. In *2nd Annual Sunningdale Accreditor's Conference*. RMCS Shrivenham, 2001.
27. C. L.Robinson and K.J.Hughes. Managing infosec risk in complex projects. In *4th Annual Systems Engineering for Defence Conference*. RMCS Shrivenham, 5-16th February 2001.
28. R. Milner. Calculi for synchrony and asynchrony. *Theoret. Comp. Sci.*, 25(3):267–310, 1983.
29. R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.
30. R. Milner. *Communication systems and the π-calculus*. Cambridge University Press, 1999.
31. Brian Monahan. From security protocols to systems security: Making a case for systems security modelling. Technical report, Hewlett-Packard Laboratories, 2003. HPL-2003-147.
32. Brian Monahan. Infrastructure security modelling for utility computing. Technical report, Hewlett-Packard Laboratories, 2005. HPL-2005-4.
33. Mark D. Ryan, Nan Zhang, and Dimitar Guelev. Evaluating access control policies through model checking. In *Eighth Information Security Conference (ISC '05)*, LNCS, 2005.
34. P. O'Hearn. Resources, concurrency, and local reasoning. *Theoret. Comp. Sci.*, 2005.
35. P.W. O'Hearn. On Bunched Typing. *J. Functional Programming*, 13(4):747–796, 2003.
36. P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
37. David Pym and Chris Tofts. A calculus and logic of resources and processes. Technical report, Hewlett-Packard Laboratories, 2004. HPL-2004-170R1.
38. D.J. Pym. *The Semantics and Proof Theory of the Logic of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002. Errata and Remarks at: `http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf`.
39. D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of **BI**. *Theoretical Computer Science*, 315(1):257–305, 2004. Erratum: p. 285, l. -12: ", for some $P', Q \equiv P; P'$ " should be "$P \vdash Q$".
40. J.C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proc. LICS '02*, pages 55–74. IEEE Computer Society Press, 2002.
41. Colin Stirling. *Modal and Temporal Properties of Processes*. Springer Verlag, 2001.
42. Richard Taylor and Chris Tofts. Modelling, Myth vs Reality, Map vs Territory. Technical Report HPL-2003-246, HP Laboratories, Bristol, 2003.
43. C. Tofts. Efficiently modelling resource in a process algebra. Technical Report HPL-2003-181, HP Laboratories, Bristol, 2003.

A A Brief Guide to Demos 2000

In reality, **Demos 2000** is two things — first of all, it is a semantically justified [7, 8] discrete-event systems modelling language and, secondly, it is a simulation based environment to support the examination and exploration of systems so described.

The **Demos 2000** 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 **Demos 2000** tend to be high-level, pleasingly short and to the point. The modelling philosophy thus 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 present a brief 'taste' of a typical **Demos 2000** definition of a system. Although not syntactically mandated in any sense, as a general rule **Demos 2000** programs pragmatically adopt the following standard shape:

1. Constant definitions:
   - **Demos 2000** 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;
2. Global variable definitions;
3. Resource definitions:
   - In **Demos 2000**, resources represent pure synchronisations (in the process-calculus sense) and can be claimed and released by means of `getR` and `putR` expressions;
4. Bin definitions:
   - In **Demos 2000**, bins represent synchronisable entities (note that the term 'resource' is used in the rest of the paper to encompass both the **Demos 2000** notion of 'resource' and the **Demos 2000** 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;
5. Class definitions:
   - In **Demos 2000**, 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;

6. Initial model population, and entity creation;
7. Run length control, typically a `hold` of some fixed duration;
8. The all-important `close` statement ends the simulation run.

In this form, we may regard **Demos 2000** descriptions as defining system 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 **Demos 2000** 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 **Demos 2000**) and pure guarded command languages is that the conditions have side effects, due to the assignment of resource to the active entity. Hence change of state is mediated not only by assignment to variables, but by assignment and the claim of resource, and also by entities becoming resources themselves.

Demos 2000 has been given a simple, elegant and informative semantics, abstracting away from the stochastic data collection, in the process calculi SCCS and CCS [28, 29]. It can be argued that the representation of resource in the synchronous semantics (SCCS) is superior to that in the asynchronous semantics (CCS) [43].

B Demos 2000 code for the example from § 3

Below is a **Demos 2000** description of the example discussed in § 3.

As remarked earlier, we use probabilities to capture pertinent aspects of (a) user behaviour and (b) authentication/authorisation behaviour. Our abstraction here illustrates how probability can help simplify models and to eliminate details deemed to be unnecessary - in this case, the dependencies upon user accounts and personal profiles. If later on we became interested in modelling greater detail of those aspects, we could extend our model to do so, perhaps in a suitably enhanced version of **Demos 2000**.

You will observe that a sizable chunk of the definition is essentially *superstructure*, such as defining DEMOS constants and setting up variables for auditing/monitoring behaviour; the remainder comprises class definitions specifying entity behaviour.

If we strip away all this superstructure and simplify, what we have left is an underlying process-algebraic 'skeleton' term that is close to being a minimal model for the system. Such a minimal model is easily turned into a compact state machine whose properties are directly amenable to validation, even via conventional exhaustive state exploration model-checking technology.

```
(* CSFW'06 - basic security example *)

 cons runlength = 1000;

(* Structural constants - these represent structural values *)
 cons tt  = 1;
 cons ff  = 0;


(* Standard DEMOS constants *)

 cons connectDelay  = 5;


(* Probabilistic sim. of simplified user behaviour *)

 cons id            = puni(1, 2000);  (* ID values are also credential values *)
 cons query         = puni(1, 10);    (* There are 10 types of query *)


(* Simplified probabilities of acceptance *)

 cons entryProb     = 0.95;  (* probability of entry *)
 cons queryProb     = 0.2;   (* probability of query acceptance *)


(* Using probability to sim. effect of authentication *)

 cons authEntryTest = binom(1, entryProb);  (* prob. sim. of entry auth. test *)
 cons authQueryTest = binom(1, queryProb);  (* prob. sim. of query auth. test *)
```

```
(* Some variables for auditing/monitoring purposes *)

 var attempts = 0;
 var checks   = 0;
 var netEntryOK = 0;
 var netEntryFAILED = 0;
 var queries = 0;
 var queryAuthOK = 0;
 var queryAuthFAILED = 0;


(* Resource bins for synchronisations *)

 bin(netGate, 0);
 bin(authCheckEntryReq, 0);
 bin(authCheckDBQueryReq, 0);
 bin(corpDBReq, 0);


(* Classes defining entity behaviour *)

 class customerRequests =
 { entity(C, customerRequests, connectDelay);
   putVB(netGate, [id, query]);
   attempts := attempts + 1;
 }

 class networkGateway =
 { local var cur_id = 0;
   local var cur_query = 0;
   local var cur_valid = 0;

   repeat{
     try [getVB(netGate, [cur_id, cur_query], true)] then {
       checks := checks + 1;
       syncV(authCheckEntryReq, [cur_id], [cur_valid]);
       hold(1);
       try [cur_valid == ff] then {
         trace("Entry to network denied to customer %v", cur_id);
         netEntryFAILED := netEntryFAILED + 1;
       }
       etry [] then {
         putVB(corpDBReq, [cur_id, cur_query]);
         netEntryOK := netEntryOK + 1;
       }
     }
   }
 }
```

```
class authDBserver =
{ local var cur_id = 0;
  local var cur_query = 0;
  local var cur_status = 0;

  repeat{
    cur_status := 0;
    try [ getSv(authCheckEntryReq, [cur_id], true) ] then {
      cur_status := authEntryTest;
      hold(1);
      putSv(authCheckEntryReq, [cur_status]);
    }
    etry [ getSv(authCheckDBQueryReq, [cur_id, cur_query], true) ] then {
      cur_status := authQueryTest;
      hold(1);
      putSv(authCheckDBQueryReq, [cur_status]);
    }
  }
}

class corpDBserver =
{ local var cur_id = 0;
  local var cur_query = 0;
  local var cur_valid = 0;

  repeat{
    try [ getVB(corpDBReq, [cur_id, cur_query], true) ] then {
      queries := queries + 1;
      syncV(authCheckDBQueryReq, [cur_id, cur_query], [cur_valid]);
      hold(1);

      try [cur_valid == ff] then {
        trace("*** Auth. Query *FAILED* for customer %v and query %v",
              cur_id, cur_query);
        queryAuthFAILED := queryAuthFAILED + 1;
      }
      etry [] then {
        trace("--- Auth. Query ok for customer %v and query %v",
              cur_id, cur_query);
        queryAuthOK := queryAuthOK + 1;
      }
    }
  }
}


(* Initial entity creation *)

entity(C,    customerRequests, 0);
entity(NG,   networkGateway, 0);
```

```
 entity(AADB, authDBserver, 0);
 entity(CDB,  corpDBserver, 0);


(* Simulation run-length control *)

 hold(runlength);


(* Final output of auditing variables *)

 trace("attempts        = %v", attempts);

 trace("checks          = %v", checks);
 trace("netEntryFAILED  = %v", netEntryFAILED);
 trace("netEntryOK      = %v", netEntryOK);

 trace("queries         = %v", queries);
 trace("queryAuthFAILED = %v", queryAuthFAILED);
 trace("queryAuthOK     = %v", queryAuthOK);

 close; (* Simulation ends *)
```