An Event-Based Paradigm for E-commerce Application Specification and Execution

Alan S. Abrahams*, David M. Eyers†, and Jean M. Bacon†
* *Department of Operations and Information Management,*
The Wharton School, University of Pennsylvania
500 Jon M. Huntsman Hall, Philadelphia, PA 19104-6340, USA
† *University of Cambridge Computer Laboratory, William Gates Building,*
15 JJ Thomson Avenue, Cambridge, CB3 0FD, United Kingdom
asa28@wharton.upenn.edu, {David.Eyers, Jean.Bacon}@cl.cam.ac.uk

Abstract

This paper describes a new approach to e-commerce application development which forsakes the traditional conceptual notions of object-orientation and exploits instead a purist event-centric approach that is felt to promote easier maintenance and implementation of specifications (policy). The core concepts of the event-centric approach are described, and the approach is compared to object-orientation as well as to existing event-based and policy-based mechanisms. E-commerce examples are provided.

1 Introduction

Business users need to be able to specify policy (specifications, processes, practices, rules, requirements, standards, and laws) in an intuitive and understandable form. They must be able to read and manage distributed and diverse business policies. Currently, policies are intermingled between object-oriented code (e.g. Java, C++, Smalltalk) and data access languages (e.g. SQL, OQL). Policies (business rules) are hidden and implicit in code and are haphazardly distributed. This makes it difficult to extract, view, and maintain business rules [4, 10]. Traditional object-oriented and procedural approaches to developing applications separate specifications (policies) from code; this may lead to inconsistencies. Peter Coad[1] comments that maintaining system requirements and source code in synchronization could 'change the very nature of software development'. Conventional approaches for this, including those of Coad's *TogetherSoft*, involve object-oriented code generation. We believe this is sub-optimal. Instead we propose a purely event-based paradigm which does not separate specifications (policy) from code.

We are developing a purely event-based architecture to facilitate the executable specification of e-commerce applications. In our approach, specifications (policies) are decomposed into events which, like operational events, are stored and animated in an event store. In *Section 2* we define the notion of events; in *Section 3* the types of events are explained. *Section 4* compares the purist event-oriented approach to the object-oriented paradigm. *Sections 5* and *6* provide guidance on exposing events and temporal relationships between events in English specifications. The interrelationship between events and policy is described in *Section 7*. We conclude with a comparison to related work and a summary of the benefits of the purist event-centric paradigm.

2 Events

The event-centric paradigm treats events as the primary abstraction. **Events** are regarded as any occurrence, process, wilful action or activity, or state, and can be identified, inter alia, from *verbs*[2]. Events may have **parameters** such as time or interval of occurrence and place (location), and events must relate to **referents**, which participate in the event. Referents are simply concepts denoted (or denotable) by a unique identifier, and may be entities (e.g. things, places, or roles) or may be other events. Referents are **participants** in the event and can be bound to the event in a finite set of **roles**. For example, the sentence 'John purchases sweets' can be represented in a purist event-centric language as 'one referent *named* John participates as purchaser (role) in an event *classified* as a purchase, with one or more referents *classified* as sweets participating as the purchased items (role)'. Naming and classification events will be describe in further detail later. Events may take other events as participants, in which case one event is said to *subordinate* another. For example '*authorisation* to *read* the

[1] Quoted at http://www.togethersoft.com/. 1 September 2000.

[2] Although events are referred to by verb name for convenience throughout this document, events are more accurately referred to by verb sense identifiers as any given verb (i.e. word or symbol) may have multiple senses (i.e. meanings). Verb senses can often be identified via consultation of a lexicon such as WordNet (http://www.cogsci.princeton.edu/~wn/) - an approach demonstrated in COLOR-X [5]. Event *occurrences* (i.e. referents that are events) are actually instances of *event types* (*verb senses*), just as activity-occurrences are instances of activities as in the *Process Specification Language (PSL)* specification [17].

file' comprises an 'authorise' event which subordinates a 'read' event. Events may also be linked to 'start', 'suspend', 'resume', and 'stop' events.

Linguists have traditionally described participants in events in terms of *semantic or thematic roles* (participant types) such as *agent* (*undertakes* the event), *patient* (*undergoes* the event), *instrument* (the means *used* to achieve the event), and *beneficiary* (directly benefits from the event). While thematic roles are often helpful and are commonly used [1, 9, 16, 20, 21], they have been criticized as the thematic role of a participant in an event may be difficult to determine or ambiguous [7]. We therefore supplement thematic roles with *domain-specific roles*. For instance, in 'John buys from Walmart' it is unclear whether John or Walmart is the agent because both are active parties: John undertakes the buying but Walmart undertakes the selling[3]. In this case using the domain-specific roles 'buyer' and 'seller' is preferable to using the thematic role 'agent' which is ambiguous. Domain-specific roles (such as buyer, lessor, lessee, participant, analyst) can usually[4] be constructed in English through appropriate suffixes such as -er, –or, -ee, -ed, -ant, -ist, or -yst appended to the verb[5].

3 Types of Events

Various types of events can be identified. These can broadly be classified as descriptive, prescriptive, factual, or assumptive events. These are explained in the following sections.

3.1 *Descriptive* events: Classification and Naming Events

Descriptive events (descriptions) are declarative in nature and include *classification* and *naming* events. Naming is a common event which recurs frequently in specifications. Proper nouns, which name a specific referent, imply naming events. The basic form of a naming event is:

```
[referent] is named [symbol/name]
```

[3] 'Buy' and 'sell' are lexical doublets that refer to the same event. 'Lexical doublets' is a term adopted from [9] that describe different lexicalisations of the same event as viewed from different perspectives. Another example of lexical doublets is 'lend' and 'borrow'.

[4] In the case of irregular verbs these rules for denoting roles cannot be used but the role name can nevertheless be easily found in a lexicon.

[5] Or more accurately, to verbal lexicalisations of the event.

Referents which are similar in some respect may be grouped by classification (categorization) events. Classification events allow us to:

- create *classification hierarchies*. For example, "A customer is a person" means "A referent classified as a customer **must** be classified as a person".
- create *sets* or *named groups* of referents. These classifications associate a **name** with items that comply with a **description** or **set of criteria**. For example:

Name: Wealthy Londoners
are classified as
Description: People with yearly income > £100k per year and telephone number beginning with 0207 or 0208.[6]

The party making the classification is implicit but may become relevant when diverse schemas need to be federated. The basic form of a classification event is:

```
[referents] are classified as
[type/kind/class/classification/category[7]]
```

Referents may be multiply typed (classified) and types (classification events) may be mutually exclusive. Many events are followed by classification events that classify the referent as being in a certain **state** – for example an 'approve' event may change an application from being classified as 'pending' to being classified as 'approved'. More complex forms of classification event also exist. For example, classification of something as 'able', 'ready', 'the same as', 'different from', or 'a member of' is often with respect to some other (extensionally or intensionally defined) referents. The form of such complex classification events is:

```
[referents] are classified as
able-to/ready-to/the-same-as/different-from/a-
member-of/etc.
[referent(s)]
```

[6] This definition would more appropriately be atomized into separate definitions for 'wealthy' and 'Londoner'. Also, more accurately, wealth is relative to some subjective benchmark.

[7] Type (synonyms: kind/class/category) is a semantic role peculiar to classification events.

Type-determination in the event-centric paradigm is supported through the triggering of a classification event when a relevant composite event is detected. In the event-centric development paradigm, the type of a concept is determined by events which the (extensionally or intensionally) identified referents have enacted or could enact (as *willing* actors or *controlled* instruments) or to which they have been, or could be, subject (as *voluntary or involuntary* patients). So, the event-centric paradigm supports at least two modes of type determination:

1. Type determination based on *operational event-history*.
2. Type determination based on *pattern of permissible future invocations* (derivable from *normative*[8] *event history*).

The **behaviour** of the type – its reaction to events – is determined by what events *must* or *can* be inserted into the event store (or what external operations must or can be invoked) for referents of that type. Events themselves are typed: typing of events can be achieved by linking the event to one or more classification events.

3.2 *Prescriptive* Events: Constraints and Prescriptions

Prescriptive events (constraints and prescriptions) are implemented using constraining *quantification events*, imperative *normative events* and declarative *modal events*.

Quantification events may specify or constrain the number of referents that participate in an event. For example 'Students may study up to 4 courses' specifies that 'in an event classified as studying, in which students participate as the actors, a maximum of 4 referents may participate as patient of the event for any given student'. Quantification events may specify exact quantities or upper or lower bounds (ranges or intervals). These may be directly specified (e.g. '4') or run-time determinable (e.g. 'the course-load limit'). Care must be taken to distinguish between *collective* ('the five lecturers together teach ten courses') and *distributive* ('the five lecturers each teach ten courses') readings of the sentence to avoid ambiguity.

The closed set of **normative events** we provide includes authorise, forbid, and oblige[9]. In all cases of permission, obligation, and forbiddance, the system may *prevent* violations, or may *detect* and act upon violations. Authorisations (permissions) and forbiddances (prohibitions) associated with a role are the **privileges** of the role – e.g. what the referent *can* do. Obligations (oblige events) associated with a named role are the **responsibilities** of the role – e.g. what the referent *must* do. Obligations may be immediate or bounded by a start and/or end -time or -event. **Rights** are taken in the broad business sense to include not only legal abilities (*powers*), but also obligations in the referents favour (i.e. *entitlements*). For example, a buyer has the *right* to receive the product bought because the seller is obliged to deliver the product bought to the buyer (i.e. the obligation is in favour of the buyer – the buyer is the *beneficiary* of the obligation). Obligations are imperative in nature. Examples of obligations in e-commerce include:

- interest and capital repayments on loans: obligation to pay
- sale: obligation of seller to deliver, obligation of buyer to pay (commitment to expenditure)
- lease payments: obligation to pay
- rebates: obligation to give discount when rebate ticket presented
- warranties and guarantees: A warranty (or warrant or guarantee) can be generalized as a *contingent obligation*: that is, an obligation that arises (i.e. is triggered) upon occurrence of certain events (contingencies). Warranties and guarantees generally have a lifespan. Examples of warranties are:
  - A 5-year product warranty may trigger an obligation to repair at no charge if the product is damaged during normal use within 5-years from the date of purchase (i.e. while it is 'under guarantee').
  - A profit warranty, as part of a share sale transaction, warrants specified profits during a certain financial period and the amount of the purchase consideration is contingent on those profits being achieved.
- pension obligations: obligation to pay regularly after retirement
- accrual of entitlements by employees, customers, or suppliers: for example, obligation to pay employees outstanding leave pay within 30 days of date of resignation from the firm
- obtaining qualifications: A referent may be obliged to fulfill specific requirements in order to obtain a qualification.

Obligations, like other events, are often *contingent*: they (i.e. 'oblige' events) may be triggered by some uncertain future events. Contingent

[8] Normative events are defined in Section 3.2.

[9] 'Authorise' is synonymous with 'permit' and 'allow'; 'forbid' is synonymous with 'prohibit' and 'disallow', and 'oblige' is synonymous with 'command'.

obligations in business practice rules may, along with other financial events, be used in the construction of financial statements.

Modal events allow the formulation of descriptive (declarative) rules which describe the capabilities of referents as well as what is possible in the current domain or environment. The closed set of *modal events* provided includes 'classify as able to' and 'classify as possible'.

3.3 *Factual* Events

Factual events are events that have occurred or are certain to occur[10]. Factual events include *user-interface* events as well as *business* events. Business events include *contractual* events which entail the incurrence of rights and responsibilities, as well as *workflow* events which entail the fulfilment of responsibilities or the uptake of opportunities. Examples of common factual events in e-commerce applications are illustrated in *Table 1*.

Event Category		Examples of events (Verb lexicalization | Deverbative noun[11] lexicalization)
Factual	**User-interface**	HTML: Clicks | Click, Selects | Select*ion*, Enters | Entry, Submits | Submiss*ion*, Displays | Display HTTP: Requests | Request, Responds | Response, SMTP (email): Queues, Sends, Receives
	Contractual	Buys | Purchase, Leases | Lease, Rents | Rent*al*, Insures | Insur*ance*, Subscribes | Subscript*ion*
	Workflow	Charges | Charge, Pays | Pay*ment*, Fulfils | Fulfil*ment*, Registers | Registrat*ion*, Signs | Signature, Verifies | Verificat*ion*, Validates | Validat*ion*, Approves | Approv*al*, Rejects | Reject*ion*

Table 1: Examples of common factual events in e-commerce applications

Contractual events (contracts) deserve further explanation. Contracts are comprised of descriptive and prescriptive events. E-commerce contracts incorporate definitions (descriptive events) and rights and responsibilities (prescriptive events), and set forth the obligations, authorizations, and legal powers of the parties to the contract. The parties to the contractual event fill named **roles**, which describe the nature of their participation in the event and may be used to refer to the parties for convenience. **Rights** are authorisations, legal abilities (powers), and obligations in a party's favour. **Responsibilities** are obligations that a party has incurred or forbiddances that it is subject to. For example the legal implications of 'buyer purchases goods from seller' are:

- Buyer is legally *obliged* to pay for goods (e.g. within x days)
- Seller is *obliged* to deliver goods (e.g. within x days of payment)
- Buyer *owns* goods upon payment. Seller no longer *owns* goods at that stage. ('Owns' is a stative event[12].)

Here we see that:

- The primary **roles** in the event are 'buyer', 'purchased items', and 'seller'.
- The main **responsibilities** represented are: that of the buyer to pay and of the seller to deliver, contingent upon payment.
- The **rights** are: of the seller to receive payment, of the buyer to receive goods contingent upon payment, and of the buyer to be regarded as owner contingent upon payment.

Contracts come into force when the terms (i.e. rules) of the contract are agreed to (upon *legal agreement*[13] events).

3.4 *Assumptive* Events

Assumptive events are events that might not actually have occurred, but are assumed to have occurred or are anticipated to occur in the future. Assumptive events are so named as they are subordinated to 'assume' or 'predict' events or the like. In stable, non-random environments assumptive events may be predicted using probability theory. Traditional event service have been hamstrung by the need to either block for delayed events (synchronous consumption), or to continue - perhaps after a short timeout - by ignoring possible delayed events (asynchronous consumption) [19]. The former course of action impedes performance whilst the latter leads to blind reactions that recklessly disregard past experience. Assumptive events are useful when rapid, semi-informed action is preferred, as they allow the event monitoring service to act quickly upon likely events. It is important that any reactions undertaken be

[10] As for all events, the *occurrence* of the event, must be distinguished from the act of *capturing* the event.

[11] Deverbative nouns are explained in 5.1

[12] Stative events are described later, in Section 4.

[13] Note that agreements are not necessarily legal agreements, so we have explicitly qualified 'agreements' as 'legal agreements'.

reversible though as the assumption may prove to be invalid and backtracking may later be required. *Figure 1* illustrates the difference between a traditional event monitor[14] and the novel notion of a probabilistic event predictor, which fires 'predict', 'assume', 'anticipate', or 'expect' events.

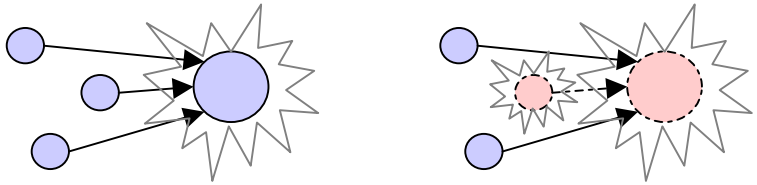

Figure 1: Traditional event monitors (left) await actual occurrences before firing composite events. Event predictors (right) may project and fire assumptive events which in turn may trigger composite events

Table 2 below summarizes the common events that occur in policy (business rule) specifications and in operational environments.

	Event (Active verb)	Name (Deverbative noun)	
Policy-related Events (Events in Business Rules)	**Descriptive (Describes)**		
	Names	Naming	
	Classifies	Classification	
	Prescriptive (Prescribes)		
	Quantify	Quantification	
	Permits, Allows, Authorizes	Permission, Authorization (*Powers*)	**Rights and responsibilities**
	Obliges	Obligation	
	Forbids, Prohibits, Denies	Forbiddance (Negative Authorization)	
Operational Events (Factual and Assumptive)	Occurs, Happens	Occurrence	
	Captures, Records	Capture, Record	
	Predicts, Assumes, Anticipates, Expects	Prediction, Assumption, Anticipation, Expectation	

Table 2: Common policy-related and operational events

[14] Traditional event monitors are described in Section 8 (Related Work).

4 Event-Centric vs Object-Oriented

The event-centric approach does not make use of the traditional object-oriented notion of objects having attributes, (which determine their state) and methods (which describe their behaviour). Instead, uniquely identified *referents* replace the notion of objects. Referents may be role-players in events such as classification events (specifying the *type* of the referent), normative events (specifying what the referent *can* and *must* do), and others. **State** is determined by the currently applicable event bindings to a referent. Constraints on state transitions are modelled as conditional forbiddances on classifications. Because events lead to (reversible or irreversible) states, State Transition Diagrams can be automatically generated from event models [5]. Traditionally some states are hard to name; the event-centric paradigm alleviates this as a state can be described in terms of the previous events that the referent has participated in, much as is done in natural language through the use of that-, who- and which-clauses. In the specification 'only *registered* and *approved* customers can buy goods' the policy can be interpreted as 'customers who have participated in a *register* event and an *approve* event – but not participated in any events that might reverse the registration or approval – are permitted to buy goods'. **Behaviour** of a referent is determined by the *ability* to, *authorization* to, or *obligation* to:

- insert new events bound to the referent into actionable queues when composite events are detected, or
- invoke external operations.

These *abilities*, *authorizations* and *obligations* are the *capabilities* and *responsibilities* of the referent, respectively. All conceptual notions traditionally associated with static object models, such as classes and relations, are discarded in favour of modelling these concepts as events. This permits the conceptual model (traditionally considered as static) to be dynamic and queryable as events can be added to and read from the event store at any time. The object-oriented notion of *classes* can be modelled in the event paradigm through *classification* events which determine the class to which the referent belongs. The referent may be reclassified multiple times during its lifetime and (arbitrarily complex patterns of) classification events may be mutually exclusive to allow the implementation of disjoint types. The object-oriented notion of *relations* can be modelled in the event paradigm simply through events, because participants in events are naturally related by the event occurrence itself. For example, in 'customers can *purchase* goods from companies', we have referents classified as

customers potentially related to referents classified as goods and companies through 'purchase' events. *Part-of* (aggregation) relations in object-oriented notations are modelled through stative events like 'has', 'owns', 'possesses', 'contains', 'includes', 'excludes', and 'involves'. Stative events differ from active events in that stative events do not involve activity, but rather refer to a current state-of-being.

5 Exposing events in English specifications

Events may be exposed from verbs in English specifications. However, verbs are not the only lexicalisations of events, and events may be lexicalised, inter alia, as deverbative nouns, modals, adjectives, nouns, and cardinals. This section describes some basic techniques for exposing events in English specifications of e-commerce applications.

5.1 Events exposed by deverbative nouns

Deverbative nouns are noun forms of verbs, and imply the existence of the event corresponding with that verb. Deverbative nouns can be detected in English specifications through indicative suffixes (such as -ion, -ment, -ent, -ure, -ance, -ence, -ancy, -ency, -ing, -ness, -al, or -y). For example: `subscription` implies a `subscribe` event, `government` may imply a `govern` event, `failure` implies a `fails` event, and `approval` implies an `approves` event.

5.2 Events exposed by modals

Modals (e.g. modal auxiliaries and suffixes) indicate the existence of modal or normative events. In English, modal auxiliaries include lexical items like 'can', 'must', 'have to', 'should', and 'will'. Certain suffixes (e.g. -able and -ible) are also normative or modal. For example, one (normative) sense of 'readable' implies '[actor] `authorises` to read'; for 'interruptible' there is an implied '[actor] `authorises` to interrupt'. Another (modal classification) sense of 'readable' is '[actor] is `classified as able` to be read [patient]' or 'reading of [patient] by [actor] is `classified as possible`'. 'Payable' may imply that a referent is `obliged` to pay; e.g. 'the account is payable' means 'the account holder is `obliged` to pay the account'.

Table 3 below illustrates how the common semantics of the English modals can be defined in terms of the prescriptive (*normative* or *modal classification* events) events defined in Section 3.2, or in terms of *detection* and *induction*. Note that, in any given sentence, a modal can have one or more semantics: that is, it is possible for a modal to simultaneously have multiple meanings. This is why English modals are considered vague. This vagueness can be avoided in event-centric specifications simply by specifying all the intended semantics of an occurrence of the modal by, for instance, choosing the relevant normative and modal classification events from a list.

5.3 Events exposed by nouns and modifiers

Nouns and *modifiers* - such as adjectives, adverbs, adjectival, and adverbial phrases - may imply classification or naming events. For example 'rich (adj.) staff (n.)' implies 'referents *classified* as staff and rich'. Alternatively, 'rich' may be relative to some subjective benchmark that is set for staff: so 'rich' staff may not be as wealthy as 'rich' basketball players, since 'rich' is used relativistically in each case.

5.4 Exposing quantification events

`Quantification` events may be implied by various language features such as *cardinals numbers* (e.g. '3') and *units of measure* (e.g. '3 metres'), as well as words and phrases such as a/one, none, no, some, few, multiple, many, most, only, at least, at most, and others.

Modal Auxiliary or Suffix	Possible Event Semantics
Can, May (not)	▪ ... `permits` (`forbids`) ... to [event*] * Note that all events/actions here are possibly parameterised (i.e. complex events/actions) ▪ ... `classifies` ... as (un)able to [event] ▪ ... `classifies` [event] as (im)possible
Could, Might	▪ ... `permits` ... to [event] **if** ... (i.e. *conditional/contingent* permission) ▪ ... `classifies` as able to [event] **if** ... (i.e. *conditional* ability) ▪ ... `classifies` [event] as possible **if** ... (i.e. *conditional* possibility) ▪ past tense of 'can' (i.e. denotes `permit` or `classify` events that have occurred before the utterance or usage of the policy)

Must, Shall, Should, Ought, Will, Has-to / Have-to, Ensure that, Needs to	▪ … `obliges` … to [event] ▪ … `classifies` [event] as necessary ▪ The system must `verify` … that [event] has happened (i.e. obliged *detection*) ▪ The system must `deduce` … that [event] has happened (i.e. obliged *induction*[15])
Should have	▪ past tense of 'must'
Must / May / Shall / Should / Ought / Will not	▪ … `forbids` … from [event] ▪ The system must `verify` that [event] has not happened
Should, Would	▪ … `obliges` … to [event] **if** … (i.e. conditional obligation)
-able, -ible	▪ … `permits` … to [event] ▪ … `obliges` … to [event]
Does, is (factual modality)	▪ … `happens / occurs` …

Table 3: Event Semantics of Modal Auxiliaries and Suffixes in English

6 Temporal Relationships Between Event

Traditional event monitors provide a limited set of temporal operators for *detection* of the relation in time between events. These operators are used solely in interrogative mood in event monitors, and aside from the notable exception of GEM [15], cannot be used for event *actuation*. *Figure 2* illustrates the difference between an event monitor and an event actuator.

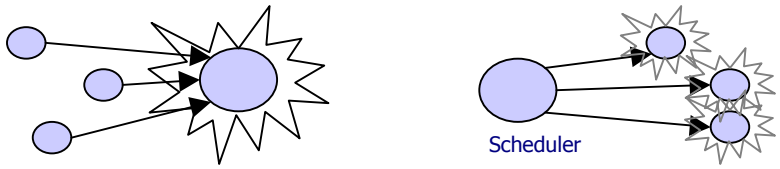

Figure 2: Traditional event monitors (left) provide only events detection. Sophisticated event actuation capabilities (right) are needed for workflow applications.

[15] Obliged induction relates to **necessity** (i.e. something that is **necessarily true**). This is the 'NEC' relationship in COLOR-X [5].

We propose the use of temporal operators in both interrogative and *imperative* mood; that is, that they should be used not only for *detecting* patterns of events, but also for *triggering* events in desired patterns, so as to control system behaviour as well as workflow. In short, temporal operators should be used to *determine* - that is *detect/ascertain* and *cause/bring-about* - temporal relationships between events. *Table 4* below lists some common English words for expressing temporal relations between events.

before, prior to, previous, pre-, earlier, precedes, past, in advance of, former, formerly, once[16], directly before, already	after, once, upon, when, whenever, if, if ever, post-, then, later, subsequent, succeeds, future, next, directly after, yet, and	during, while, when, on, in, at, throughout, and	past, last, within, next, passed / elapsed as at, until, up until	ordinals: first, n-th, last, n-th from last

Table 4: Common words (and affixes) for expressing temporal relations between events

Table 5 demonstrates how tenses in English sentences may determine the temporal relationships between the event of *utterance* or *usage* (interpretation) of a sentence in a specification, and the events *reported* in the sentence. *Figure 3* illustrates the relationship between a 'purchase' event and the utterance event in 'I purchased goods'.

Tense	Common Implications for Temporal Relations Between Events
PAST	Start of reported event is **before** utterance/usage
PAST PERFECT	Start and end of reported event are **before** utterance/usage
PRESENT	Start of reported event is **before** utterance/usage. End is unknown or soon in the future.
PROGRESSIVE	Start of reported event is **before** utterance/usage. End of event is **after** utterance/usage.
FUTURE	Start of reported event is **after** utterance/usage

[16] 'Once' has multiple meanings: Amongst these are 'at one time in the past', but 'once' can also mean 'after' (e.g. 'once he has submitted the form, notify…').

Table 5: Tenses and temporal relations between reported events and utterance/usage events

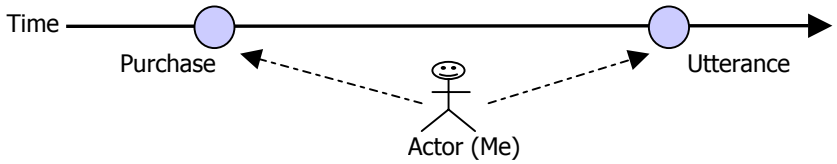

Figure 3: Relationship between the utterance event and the reported 'purchase' event in 'I purchas*ed* goods' (past tense)

Note that time relations are not the only relationships between events. Additional operators, such as event-participant and event-parameter matching, comparison, substitution, and constraint, can be provided to determine – i.e. ascertain or bring about – other non-time relationships between events.

7 The Interrelationship Between Events and Business Rules

The term 'business rules' in synonymous with the term 'policy'. Policy defines the system requirements (specifications) and how they are implemented. Policy describes how items are *named* and *classified* in the system (*descriptive* policy) and what events *can* and *must* be done to items or sets of items (*prescriptive* policy). Policies are used to describe how a system should operate [10] and can therefore be used for requirements specification and animation. Policy-related events occur across diverse specifications in various application domains and include those for classification, authorization, forbiddance, and obligation. Policies have **domains** (which describes which referents the policy applies to, in various **roles**), a **modal or normative event** (e.g. authorize, oblige, or forbid), an **event subordinated to the modal event**, and optional **conditions**. For example in 'Customers can purchase goods if they are registered':

- the **subordinated event** is 'purchase'
- as indicated by the modal 'can', the **modal (i.e. subordinating or policy) event** is 'authorize'[17]
- the **domains** are:
  - Referents classified as customers, acting in the 'authorized' and 'purchaser' **roles**.
  - Referents classified as goods, acting in the 'purchased' **roles**.
  - Implicit referents acting in the 'authorizer' roles
- the **condition** (i.e. the circumstance under which the 'authorize' event is triggered) is 'the aforementioned referents classified as customers have also participated in a register/registration event'.

The policy can therefore be understood as 'the participation of referents *classified* as customers in a register event leads to the triggering of an *authorization* of said referents to participate in a *purchase* event where referents *classified* as goods are purchased'. It is clear that complex access control policies, more expressive than traditional Access Control Lists (ACLs) can be specified using this formalism, as can complex obligation policies.

The policy life-cycle can be described in terms of events. [22] explains that the policy life-cycle consists of: policy definition (creation), policy transformation (high-level policies transformed into low-level policies), policy activation, policy monitoring, policy pausing and resumption, policy change or adaptation, policy enforcement, policy deactivation, and policy deletion. We supplement these notions with the following additions:

- We distinguish between policy *approval* events and policy *application* events as approval and activation are separate events. The window of applicability of a policy (i.e. the interval between the policy 'applying' and 'ceasing to apply') may have an upper and lower bound, each of which may be delimited by a date-time stamp or an event. For example, a policy may be applied directly after the approval event of that policy, or application of the policy may be deferred to a later time or event.
- We term policy transformation as policy *elaboration* as we feel this is more descriptive. Policy elaboration involves linking high-level policies (i.e. the 'what') to the lower-level policies that enforce or implement that policy (i.e. the 'how'). This may occur at multiple levels.
- During monitoring, a policy may be *violated* or *complied with*. Actions may *violate* or *comply with* policy, or their status may be *unknown* if they have not yet been assessed against policy. When a policy is violated a *violation* event is signalled, whose participant is the policy being violated. A violation may be *respond*ed to by the system or by users. The violation may be *resolved* or *outstanding* (unresolved).

[17] 'Can' may have other senses aside from authorization (permission) – for example, it may indicate legal ability (legal power).

- We distinguish a number of ways in which a policy may *cease* to apply (i.e. be deactivated). For example, policies may *lapse* or be *revoked* by an empowered 3rd party.
- We allow for policies to be *grouped* into ('classified as being a member of') *packets* for manageability. If, for instance, a packet is deactivated, all policies within that packet are regarded as deactivated.
- Support for conflict. If several rules meet their conditions at the same time, we provide for a determination of the rule with the highest precedence, which may *suppress* other conflicting or non-conflicting rules from firing (i.e. being enforced).

Table 6 below demonstrates the events and states that pertain to the business rules (policy and packet) life cycle.

Policies or packet events	Policy or packet states
Policy or packet created / defined	Policy or packet exists (but not yet active)
Policy elaborated	
Policy classified as member of (grouped into / assigned to) packet	
Policy or packet changed e.g. permission transferred	
Policy or packet approved	
Policy or packet applied / activated. In the case of permissions this is referred to as granting.	Policy or packet active / enforced / in force
Policy or packet paused	Policy or packet inactive
Policy violated (Creates a violation event)	
Policy or packet resumed	
Policy or packet de-applied / deactivated	
Policy or packet deleted	

Table 6: Business Rule (Policy and Policy Packet) Events Through the Business Rule Life Cycle

Table 7 below shows events relating to policy `violations`.

Violation events	Violation states
Trigger violation (Policy violated)	Violation outstanding
Respond to violation	
Resolve violation	Violation resolved

Table 7: Policy Violation Events

Obligations are a special type of policy and certain peculiar events apply to the life cycle of obligations. These supplementary events, which are summarized in *Table 8* below, are:

- Obligations may *arise*: that is, be *imposed* or *incurred*.
- Obligations may *fall due*. An obligation that has arisen (as a result of events) and fallen due (as a result of the passage of time or occurrence of events) must be *satisfied* within a certain time or before the 'due by' date-time or event, otherwise the obligation is *violated*.
- An obligation may be *settled* (i.e. *fulfilled*, *discharged*, or *complied-with*). The International Accounting Standards Committee Framework [14] lists a number of business events that may result in the settlement of an obligation, including: payment of cash, transfer of assets, provision of services, replacement of the obligation with another obligation, or conversion of the obligation to equity. An obligation may cease if it is completely fulfilled or may *continue* if it has been partially fulfilled.
- The obligation may *cease* to apply in a number of ways:
    - The obligation may be *completely fulfilled* (fulfilled in its entirety).
    - The obligation may be *extinquished*. Again following the terminology of [14], extinguishment of an obligation may be via the waiver or forfeiture of rights (obligation *waived*) by the obliging party, or by the removal of rights by an empowered 3rd party (obligation *cancelled*).
    - The obligation may *lapse* or *expire* as a result of the failure of another party to fulfill mutual obligations within a given time: for example, an obligation of a buyer to pay may lapse if the seller has not fulfilled their obligation to deliver within 15 days of the placement of the order.
    - The obligation may be *over-ridden* by another obligation that has precedence.

Obligation events
(Contingent) Obligation arises: incurred or imposed
Rights ceded[18] (i.e. *beneficiary* of obligation changed)
(Contingent) Obligation falls due. (Previously-incurred obligation enforced, often conditional upon circumstances).
Obligation settled / fulfilled / discharged. The obligation will cease if completely fulfilled, or may continue if partially fulfilled.
Obligation `ceases`: ▪ Obligation `settled` / `fulfilled` / `discharged` ▪ Obligation `extinguished` o Obligation `waived` / Rights `forfeited`, `surrendered`, `relinquished` or `ceded` by beneficiary o Obligation `cancelled` by empowered 3rd party ▪ Obligation `lapses` / `expires` ▪ Obligation `over-ridden`

Table 8: Supplementary Policy Events Through the Life Cycle of Obligations

As previously mentioned, contracts are named packets of descriptive and prescriptive events. Certain additional events may apply to contracts which determine their state and the state of the rules contained within them. *Table* 9 below summarizes events pertinent to contracts.

Contract events	Contract states
Contract `created`	Contract `exists`
Contract signed	
Contract `legally agreed to`	Contract `in force` / `entered`
Contract `nullified`	Contract `void`

Table 9: Supplementary Policy Packet Events Through the Life Cycle of Contracts

[18] 'Cede' has two verb senses: the first involves transferring rights, the second involves only surrendering or waiving rights.

8 Related Work

Existing **policy approaches and languages** are primarily targeted at system or network management [8, 22] or access control [3, 11, 12, 13]. The expressiveness of each approach varies. For example OASIS [11], LaSCO [12], QCM [13] and PolicyMaker [3] can express only authorization policies, whereas Ponder [8] can express authorization and obligation policies as well as some basic descriptive policies through an associated name service. All adopt a traditional object-oriented framework, above which the policy notation is laid. COLOR-X [5], though not developed as a policy notation, is a conceptual modelling framework that incorporates various policy concepts such as the 'permit' and 'must' modalities. The end-goal of COLOR-X is the generation of *object-oriented code* (classes, attributes, and methods) from a conceptual schema or functional specification, whereas we propose the generation of various types of *events* to be stored and animated in an event store.

Similarly, **commercial rules engines** like *Business Rule Solutions Inc's RuleTrack*, *Blaze Software's Advisor*, *Vision Software's Vision JADE*, *Usoft*, and *ILOG Rules* adopt an object-oriented or procedural approach. Web application servers like *ATG Dynamo*, *BEA Weblogic*, and *Microsoft Site Server 3.0* and *Commerce Server 2000* implement content personalization rules via a combination of XML (for tagging) and hooks to an existing object model. Again the rules languages are implemented above object-oriented or procedural fundamentals.

Most current **event monitoring services** [2, 6, 15, 18] focus on event detection and neglect event actuation. GEM [15] has been integrated with the Ponder policy language, and CEA has been integrated with the OASIS access control policy language. In both cases, procedural or object-oriented languages are used as the basis. These event monitoring services allow composite events to be fired. However, aside from the notable exception of GEM, these event monitors provide no actuation facilities or operators for triggering structured patterns of events as would be necessary for workflow management in e-commerce applications.

Workflow languages, as exemplified by the Process Specification Language (PSL) specification [17], allow routing policy to be defined but tend to neglect notions such as those of descriptive events and modalities that are crucial to coherent policy specification. Our event-based policy language is sufficiently expressive to model the complex *joins* and *splits* provided by workflow products:

- A join (synchronization, or rendezvous) point involves *forbidding* the next (successor) process from starting until the condition (e.g. 'and', 'or', or

complex condition) has been fulfilled. The successor process is then said to be 'released'. Some processes may be required to finish simultaneously, in which case the end of one process *obliges* the other process to end. A join is actually a DETECT-and-CONTROL mini-workflow that occurs concurrently with some other processes. Detection (monitoring) involves determining which processes have started or finished.

- A split (selection) point involves selecting and *obliging* one or more successor processes to start (perhaps simultaneously), which may involve *forbidding* one or more successor processes from starting. A split is actually a SELECT-and-CONTROL mini-workflow that occurs concurrently with some other processes.

CONTROL (in the case of DETECT-and-CONTROL and SELECT-and-CONTROL mini-workflows) involves obliging other processes to start or finish, authorizing other processes to start or finish, or forbidding other processes from starting or finishing.

9 Benefits of the Purist Event-Centric Paradigm

As was described in *Sections 3.1* and *3.2* above, descriptive and prescriptive business rules can be modelled as various types of event and stored in an event store along with operational data (factual events). Data, information and control can therefore flow and be stored and manipulated as events. For example:

- Data: 'John *earns* $100,000 per year'
- Information (Descriptive): 'User who earn above $100,000 are *classified* as wealthy'
- Control (Prescriptive): 'The system is *obliged* and *authorized* to forward submitted orders from wealthy users[19] to the priority despatch clerks'

Specifying the application using events eliminates the separation between model and code. Instead of storing business rules in database stored-procedures or triggers, or in application code, rules are stored as events in an event store. Code is stored in the event store as descriptive and prescriptive events and executed directly by detecting and firing patterns of events. Aside from basic operations for creation, modification, removal, copying, and movement of events, the event store will also provide composite event detection, event firing, and event correlation services, as well as generic services for sorting and counting and other aggregate functions. This uniform treatment of business rules and operational data is novel and allows the common event infrastructure to be used for processing both business rules and operational data. Processes (process information), data, and metadata can be seamlessly queried, altered, and managed, using a uniform infrastructure. Code can be easily introspected as it is merely stored as events; the software is highly reflective. Lastly, code is self-documenting – a natural language generator can be used to generate sentences describing each event – and is therefore highly readable and understandable for successive generations of developers. We therefore believe that the uniform treatment of business rule events and operational events may have significant advantages over storing business rules in triggers, stored procedures, or dispersed through application logic.

[19] The adjective 'wealthy' in 'wealthy users' implies a *classification* event, as does the noun 'users'. We are looking for referents *classified* as 'wealthy' (against some benchmark) and 'users'.

10 Conclusions

Policy can be used to specify and implement e-commerce application requirements. The problem of specifying and managing policy in object-oriented e-commerce applications is significant. Our solution is to dispel traditional object-oriented notions and adopt a purist event-centric paradigm. The advances beyond previous work in policy specification and event-monitoring systems have been articulated. We have illustrated the practicality of the solution by describing how events can be exposed from natural language specifications and managed in an event store. The uniformity of treatment of descriptive, prescriptive, and operational events in our solution may provide benefits including readability, manageability, maintainability, and flexibility in e-commerce application development.

11 Acknowledgements

This research was funded by grants from *Microsoft Research Cambridge*, the *Cambridge Commonwealth and Australia Trusts*, the *Overseas Research Students Scheme* (UK), and the *University of Cape Town Postgraduate Scholarships Office*.

12 References

[1] Allen J. "*Natural Language Understanding: Second Edition*". Benjamin/Cummings. pp 1-41, 248. (1995).

[2] J. Bacon, J. Bates, R. Hayton and K. Moody. "*Using Events to Build Distributed Applications*". In Proceedings of the ACM SIGOPS European Workshop. (1996).

[3] Blaze M, Feigenbaum J, and Lacy J. "Decentralized Trust Management". *In Proceedings of the IEEE Conference on Security and Privacy*. Oakland, CA. (1996).

[4] Blaze Software. Blaze Advisor Technical White Paper. (2000).

[5] Burg JFM and van de Riet RP. "*COLOR-X: Object Modeling Profits from Linguistics*". In Proceedings of the 2nd International Conference on Building and Sharing of Very Large-Scale Knowledge Bases. Enschede, Holland. (1995).

[6] Carzaniga A, Rosenblum DS, and Wolf AL. "*Design of a Scalable Event Notification Service: Interface and Architecture*". Technical Report CU-CS-863-98. Department of Computer Science, University of Colorado, Boulder. (1998).

[7] Davis T. "Lexical Semantics and Linking in the Hierarchical Lexicon". PhD Thesis. Stanford University, Department of Linguistics, pp 17-69. (1996).

[8] Damianou N, Dulay N, Lupu M, Sloman M. "*Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. The Language Specification. Version 1.11*". Imperial Collect Research Report DoC 2000/1. (2000).

[9] Expert Advisory Group on Language Engineering Standards (EAGLES). "*Thematic Roles*". Available at: http://www.ilc.pi.cnr.it/EAGLES96/synlex/node62.html. (2000).

[10] Goh, C. "A Generic Approach to Policy Description in System Management". HP Laboratories Technical Report HPL-97-82. Bristol. Available from: http://www.hpl.hp.com/techreports/97/HPL-97-82.pdf. (1997)

[11] Hayton RJ, Bacon JM, Moody K. "*Access Control in an Open Distributed Environment*". Proceedings of the IEEE Symposium on Security and Privacy. Oakland CA. pp3-14. (1998).

[12] Hoagland JA, Pandey R, Levitt KN. "*Specifying and Enforcing Policies using LaSCO: the Language for Security Constraints on Objects*". Department of Computer Science, University of California, Davis. Position Paper at the Policy Workshop, HP Laboratories, Bristol, UK. (1999).

[13] Kakkar Pankaj, McDougall M, Gunter CA, and Jim T. "*Credential Distribution with Local Autonomy*". Department of Computer and Information Science. University of Pennsylvania. (1999).

[14] International Accounting Standards Committee. "*International Accounting Standards 1999*". London, United Kingdom. (1999).

[15] Mansouri-Samani M and Sloman M. "GEM: A Generalised Event Monitoring Language for Distributed Systems". *IEE/IOP/BCS Distributed Systems Engineering Journal*. Vol. 4, No. 2. (1997).

[16] Palmer F. "*Grammar*". Penguin Books. (1984).

[17] Schlenoff C, Gruniger M, Tissot F, Valois J, Lubell J, and Lee J. "*The Process Specification Language (PSL) Overview and Version 1.0 Specification*". National Institute of Standards and Technology. (2000).

[18] Segall B, Arnold D, Boot J, Henderson M, and Phelps T. "*Content Based Routing with Elvin4*". ProceedingsAUUG2K, Canberra, Australia. (2000).

[19] Schwiderski S. "*Monitoring the Behaviour of Distributed Systems*". PhD Thesis. University of Cambridge Computer Lab. (1996).

[20] SIL International. "Semantic Roles". Available at: http://www.sil.org/linguistics/glossary/. (2000).

[21] Sowa JF. "Knowledge Representation: Logical, Philosophical, and Computational Foundations". Brooks/Cole. Pacific Grove, California, USA. (2000).

[22] Wies R. "*Using a Classification of Management Policies for Policy Specification and Policy Transformation*". Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management. Santa Barbara, California, USA. (1995).

13 Author Biographies

Alan S. Abrahams is a lecturer in Operations and Information Management at the Wharton School, University of Pennsylvania. Prior to this, he worked as a postdoctoral Research Associate at the University of Cambridge Computer Laboratory. His research interests are contract-driven application development and control, and agent simulation. He holds a PhD from the University of Cambridge, and a Bachelor of Business Science from the University of Cape Town.

David M. Eyers is currently a researcher at the University of Cambridge Computer Laboratory and a member of King's College. He is primarily working on policy-driven distributed access control systems. By the end of 2004, his work focus will shift to a direct involvement with the development of the CamPACE software system. A British EPSRC grant will support this development in collaboration with the Wharton School at the University of Pennsylvania.

Jean M. Bacon leads the Opera research group at the University of Cambridge Computer Laboratory, and is a Reader in Distributed Systems and Fellow of Jesus College Cambridge. She is author of *Concurrent Systems* and, with Tim Harris, *Operating Systems: Concurrent and distributed software design* published by Addison-Wesley. She is also Editor in Chief of the *IEEE Computer Society's Distributed Systems Online* (*DS Online*); see http://dsonline.computer.org. Her current research interests include role-based access control, business contracts, event-based middleware, and policy.