A Basis for E–Commerce Semantics: The Market Domain — Behaviour, Action and Information Semantics —

Dines Bjørner, Li Zhu, and Nicolai Dufva Nielsen*
Informatics and Mathematical Modelling
Building 322, Richard Petersens Plads
Technical University of Denmark, DK–2800 Lyngby, Denmark
E–Mail: db@it.dtu.dk, URL: http://www.it.dtu.dk/~db†

February 1, 2001

Abstract

To understand possible data semantics for E–Commerce, it is claimed, one must first try understand possible semantics for 'The Market'. We focus on the consumer market. As an example, we understand the consumer market to consist of consumer. retailer, wholesaler and producer, ie. essentially `buyer` and `seller` *behaviours*[1] — a market which is equipped with *actions* that query (for availability, price and delivery quotation), quote (ie. answering such queries), order (ie. buying, purchasing), confirm orders, deliver (ie. *sell*), accept (or reject) deliveries, invoice (accepted deliveries), payment, acknowledge, decline — as well as other functions (searching for buyers or sellers, for merchandise, etc.). Understanding the indivual ("intra") as well as the combined ("inter", event) "player" (buyer/seller) *behaviours*, as well as the *actions* they locally ("intra") engage in, leads us, finally, it is claimed, to a better understanding of the concept of *information* in "The Market". *Data semantics* follows from this — as *data* is taken to be `computerised representation` of *information*.

The main aim of this paper is to propose that the semantics of E–Commerce Systems be developed and understood (i) in terms of formal models of the domain of buying and selling (ie. trading, "the market"); (ii) in terms of possible requirements to the computing and communication support for extended such markets; (iii) in terms of possible software (in fact computing systems [hardware and software]) design: Architectures and program organsiations, for such requirements; and (iv) in terms of the relations between the three (i–ii–iii): The `domains`, the `requirements` and the `design`.

We shall primarily delve into study area (i) !

*Miss Li Zhu and Mr. Nicolai Dufva Nielsen are MSc students at the Department of Informatics and Mathematical Modelling, Technical University of Denmark.

†The ".it." in both the E–Mail and the URL may be changed in near future to ".imm.".

[1]This 'Market' is possibly extended with some regulatory agencies (Trade & Commerce Commission or the like) which we do not presently discuss.

Contents

1 Introduction

1.1 The Contribution

We believe that four main strengths of this paper are:

- The paper [1] `evolves` towards a *Data Semantics* from the basis of an *Information Semantics* that, again, emerges froma *Behaviour* and *Action Semantics;*
- the paper [2] `applies` a "proven technology", that of *Formal Semantics,* to an emerging field — very much in need of precision;
- the paper suggests that one [3] `separates the concerns` of understanding (α) the application *domain* [from before computing & communication is introduced] from understanding (β) the *requirements* [and show how they can be "derived" from a proper domain understanding], and `separates` these again from understanding (γ) the *software design;*
- and the paper [4.1] `demonstrates` how to subject the Electronic Commerce application domain to rigorous, formal analysis — and [4.2] `proposes` the framework for an *E–Commerce Domain Model.*

We submit the above as outlining one proper approach to *the software engineering development of E-Commerce systems.*

1.2 The Problem

In this introduction we focus on the problem. First through the exemplification (Sect. 1.4.2) of some "traditional market" scenarios: (Scenario I) Consumer ↔ retailer, (Scenario II) consumer ↔ retailer ↔ wholesaler, and (Scenario III) consumer ↔ retailer ↔ wholesaler ↔ producer market. We briefly discuss E-Commerce on the background of these examples.

The present section, in addition, mentions some sources of inspiration and outlines the structure of the rest of the paper.

The problem to be tackled by this paper, and its sequels[2], is the following:

- **The Problem Brief:**

  To understand some concepts of E–Commerce and E–Business in terms of a "triptych" software development paradigm and in terms of the formal models established for its parts.

  The triptych paradigm, such as we shall deploy it, starts by understanding the domain of commerce, or as we shall call it, "the market", *void of any reference to computing and communication, whether required or designed.* This is the main part of the paper. Then it proceeds to record an understanding of requirements that may be put to computer and data communication support for, in this case, such commerce, ie. for such business. The authors would like the paper to end with the drafting of an, or the, essense of possible software architectures that satisfy domain requirements, and of possible program organisations that satisfy interface and machine requirements.

  The formal models — of this paper — cover mostly the domain. But also requirements and software design models, their analysis, and the analysis of their relations, are needed in order to arrive at reasonably fuller understandings of the behaviour (flow), action (function) and information (data) semantics of E–Commerce.

[2] A sequel paper is to treat, more in-depth, relations between requirements and software design

1.3 E–Commerce as a Target for Testing our Development Paradigms

We are ourselves not very versant in the field of E–Commerce. Our reason for delving into this area is to take aim at some of the core issues of the subject (E–Commerce) and analyse them from the perspective of our triptych software development paradigm: *From domains via requirements to software,* and by means of our abstract, formal modelling (ie. specification) principles and techniques.

That is: We believe we have a sensible bouquet, perhaps not a complete such bouquet, of principles and techniques for software development — and now we wish to see whether they are sufficient to capture some essence of the E–Commerce concepts while also contributing to our understanding of those concepts. The IFIP WG 2.6 Working Conference for which this paper is being submitted offers a relevant forum for presenting and discussing our work, to find out whether we are on a relevant track, and to obtain corrections and stimuli.

1.4 Informal Preview "A Market" Concept

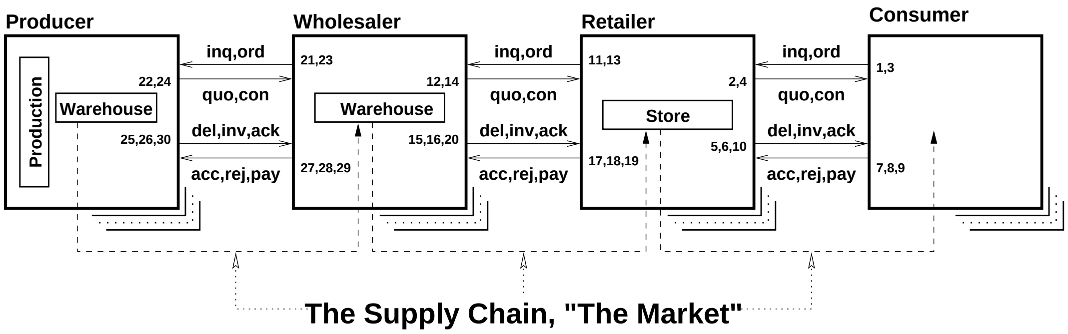
1.4.1 Informal Synopsis of "A Market" Concept

Generally: Consumers inquire about and orders goods. Orders may be simple: n items of type k goods, or may be composite: n_a items of type k_i, n_b items of type k_j, $\ldots$, n_z items of type k_m. An inquiry may lead to a quotation for the order: delivery places and times, prices and payment conditions, acceptance criteria, warranty conditions, etc. A consumer order from a retailer may be fully, partly or not immediately fulfillable from retailer stores. If not fully, then the order, or parts thereof, may be referred to a wholesaler. Wholesalers could be assumed to have no warehouses, but to bundle incoming retailer orders into batches, then ordered from producers.[3] A composite order may thus lead to inquiries across sets of retailers, and, by these, to referrals to sets of wholesalers, etc. — to producers !

1.4.2 "Conventional" Markets

The conventional market consists of the traditional supply chain. We illustrate and analyse such a supply chain market.

Figure 1: "The 'Conventional' Market"



Annotation:

[3]This restriction, "Wholesalers ... to have no warehouses", is just a "didactic device": It serves only to create some resemblance of asymmetry. To equip wholesalers with warehouses is, most likely, more actual, more realistic, but then our exploratory model would "miss" the exception, where some wholesalers do not have a local warehouse. In any case, our experimental development, as recorded in this document, is just that: Experimental. Another development, "copying" the present model, can start by adding the assumption.

- The four boxes of the diagram denote individuals or enterprises of the labelled kind.
- The triple 90^o angles (two full lines, one dashed) "shadowing" these boxes shall express that there may be any number of instances of the respective individuals and enterprises.
- The labelled arrows between boxes denote transactions (ie. events) between individuals and/or enterprises: verbal, written, or the physical transfer of goods and monies: inq queries, ord orders, quo quotations, con confirmations, del deliveries, inv invoices, acc accepts, rej rejections, pay payments, and ack acknowledgements.
- The inside "Store" box of the retailer denotes a repository of goods. Similarly for wholesaler and producer "Warehouses".
- Deliveries to consumers occur from the store, and hence an order for some goods not in store, when first ordered, is delivered, from a replenished store only after delivery to that store from a producer — via a wholesaler.

With what we now have we can outline the following possible scenarios:

- **Scenario I:** "Simplest" Market: Consumers & Retailers

  The 'simplest' seemingly "self–contained" market fragment is that of the interaction between consumers and retailers. We make it 'simple' by not considering relations to other players.

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$:

  A consumer inquires as to availability and price of some goods (inquires and obtains quotation: (1,2)), then consumer either decides not to buy, or to order these, and obtains confirmation (3,4).

  $(5,6) \rightarrow ((7,9) \mid 8) \rightarrow 10$:

  Either the order is delivered immediately, with an invoice, (5,6), as in a visit to your local grocery store, or they are sent or picked up later. Upon closer inspection the consumer either accepts the goods and pays (7,9), or rejects and returns them (8). Finally the retailer acknowledges receipt of payment or rejection (10).

- **Scenarion II:** A Simple Supply Chain: Consumers, Retailers & Wholesalers

  $1 \rightarrow 11 \rightarrow 12 \rightarrow 2$:

  The retailer does not have the queried goods in store and hence passes the inquery on to the wholesaler, who in turn sends a quote to the retailer, who (finally) responds to the consumer. Then follows the ordering process:

  $3 \rightarrow 13 \rightarrow 14 \rightarrow 4$.

  Etcetera.

- **Scenario III:** The "Full" Supply Chain: Consumers, Retailers, Wholesalers and Producers

  $1 \rightarrow 11 \rightarrow 21 \rightarrow 22 \rightarrow 12 \rightarrow 2$:

  As above, only the wholesaler does not have information on the queried merchandise and, in turn refers to the producer. The producer informs (sends a quote to) the wholesaler who, in turn informs (sends a quote to) the consumer.

  Etcetera. A gorily prolonged process might ensue:

  $3 \rightarrow 13 \rightarrow 23 \rightarrow 24 \rightarrow 14 \rightarrow 4 \rightarrow (25,26) \rightarrow ((27,29) \mid 28) \rightarrow 30 \rightarrow (15,16) \rightarrow ((17,19) \mid 18) \rightarrow 20 \rightarrow (5,6) \rightarrow ((7,9) \mid 8) \rightarrow 10$

Many other scenarios are possible.

We observe:

- Several transitions may proceed in parallel, ie. concurrently, or "out–of–expected–sequence":

- Confirmation and invoicing (4 and 6) may proceed in parallel;
- invoices may arrive before deliveries;
- deliveries from producers to wholesalers, and then from wholesalers to retailers (and possibly onwards to consumers) may be "short–circuited": Producers could deliver straight to retailers, or even consumers; wholesalers could deliver straight to consumers.
- *&c.*

• The symmetry between pairs:

  - (consumer,retailer),
  - (retailer,wholesaler), and
  - (wholesaler,producer)

  has been emphasized.

• For the sake of argument we could assume that there is an asymmetry: Retailers have a store, wholesalers may not, but have producers deliver directly to retailers.

• For the case that wholesalers also have stores, the "symmetry" is rather definite.

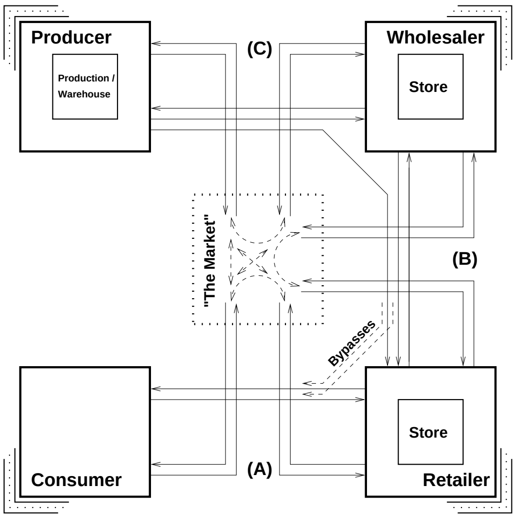
• Differences (information, but not flow, "symmetries") will be in quote strategies, wholesale versus retail prices, quantity rebates, etc.

• Symmetries can be exploited when explaining things, including formalising the four "player" processes: Retailers and wholesalers can be suitably parameterised.

1.4.3 A More General, "The E–Commerce Market"

In the more general market queries from a consumer (a retailer or a wholesaler) is directed at "the market": Some conceived totality of all retailers (respectively wholesalers and producers). We refer to Figure 2.

Figure 2: "The E–Market"



- "The E–Market" (or: "The E–Commerce Market") is a conceptual device.
- It need not be a physical, tangible phenomenon.
- "The E–Market", ideally, represents "all there is knowable" about consumers, retailers, wholesalers and producers.
- There may not, in actuality, be any individual, or any enterprise which possesses this "all knowing".
- But as a concept we can operate with it in the belief that this knowledge is somehow centralisable and representable.
- Also: The traditional, "sequential" supply chain has been "broken": We seem to allow that consumers inquire and/or order directly from wholesalers or producers, and that retailers similarly go straight to the producer.
- This, as we reveal it, emerging concept of, "E–Commerce Market" thus allows queries, even tenders for offers, and, likewise, bids, to involve any number of retailers, wholesalers and producers. (Cf. the "shadow" boxes and the unspecificity of the (A, B, C) "non-neighbour negotiations" [Figure 2 on the preceding page].)
- The "bypasses" thus illustrates "factory outlet" deliveries directly to consumers from producers, or invoicing directly from wholesalers to consumers. We leave it to the reader to "complete" the symmetries.
- The E–Commerce "Market" box can also, conveniently, be used to represent market rules & regulations as upheld by local, state and regional (ie. international trade association [EU, NAFTA, WTO]) — as well as possible consumer and trade industry — regulatory bodies.

1.5 Discussion

There are many concepts of trading in the market that we have not touched upon. Some will be revealed in our section on domain descriptions.

The problem we often see is that the three are confused:

- That much too little effort is spent on understanding the domain void of any reference to requirements to software.
- That when a claim is made that descriptions and analyses are given of the domain, it is more often descriptions and analyses of requirements to — than of — that domain.
- And that when software architecture descriptions are given they are often descriptions which involve presenting also (for a first time) some, if not all the requirements to be satisfied by that architecture.

We hope to clarify some of these issues in Sections 2—6.

We end our introductory rough sketch and part analysis of "the E–Commerce Market" concept here.

The purpose of the next section, on 'Domain Description', is to fill in necessary details — while, on the other hand, emphasizing that we do not wish to specify too much about "market" concepts as fluid as anyone sees them today !

1.6 Sources of Inspiration

The authors must, dutifully, confess that they have not studied the seemingly "vast and unstructured" literature on E–Commerce. Thus we readily admit that our contribution may be irrelevant with respect to elsewhere published papers. Suffice it, however, to record two things: That we sat in on several sessions at a recent international event listening to many paper presentations on the subject: [1, 2, 3, 4, 5, 6], and that we in particular enjoyed, and derived some not insignificant inspiration from [7].

Dillon et al., [7], in their nicely systematic treatment, introduces, the concepts of *real world, conceptual model, software structure model,* and code model — where the transitions (the commas in the list just given) correspond to *analysis, design and implementation.* The *conceptual model* of [7] corresponds, roughly to, as we call it, a domain description (Sections 3—6) of the, or an "actual world" which is then *analysed.* The *design* (transition) corresponds to our *projection, instantiation, extension* and *initialisation* resulting in what we call requirements, but which seems to be termed *software structure model* by [7]. We do not, in this paper, cover *implementation*

In that sense, by not covering requirements or design, the current paper is a torso: Only by covering the full triptych: Domain, requirements and software design, can we hope to relate the many, at times — at least as wee see them — "confused" ideas: Some representing domain concepts, some representing requirements ideas, and finally some representing software design notions.

1.7 Structure of Paper

The paper is structured as follows:

Sections 2—6 narrate and formalise a description of the domain of 'markets', void of any reference to computers and software.

- Section 2 informally outlines the conventional market — its behaviours and transactions — upon which
- section 3 bases its informal and formal descriptions of market as well as "player" behaviours. The emphasis here is on the fact that in the market any behaviour is possible — whether desirable or not — and on the fact that any "player" (buyer or seller) at any time may wish to "switch" between transactions with other players — not just pursue one sequence of such with only one other "player".
- Section 4 details the orderly, but individual behaviour — as one would perhaps like to see it in the market — of a specific pair of (hence individual) consumer and retailer behaviours.

  One can at least talk about its desirability or possibility in the market, and one can hence define it, although it may never occur in the market. One may therefore consider this section as outlining how a desirable E–Commerce market might be requirements constrained.
- Whereas Sections 3—4 dealt with overall system and "player", respectively specific "player" beaviours, Section 5 analyses specific buyer/seller actions and the information they require and generate.
- Section 6, on the background of Sections 4–5, returns to illustrate a point first broached in Section 3: That of the arbitrary and "multiplexing" behaviour of "players".

Each of these sections end with a discussiion, Subsections 2.3–6.3. The purpose of these discussions is to "return" from the extensive example to the methodological issue of the present paper: *How the informal and formal treatment of this paper can serve as a basis for understanding 'E–Commerce Semantics' and for developing 'E–Commerce Systems'.*

Thus Sections 2–6 illustrate an approach to computing systems development in which one first establish descriptions of the potential application domain, usually by setting a scope and a span which may be much larger than that of the requirements to, and the actual software.

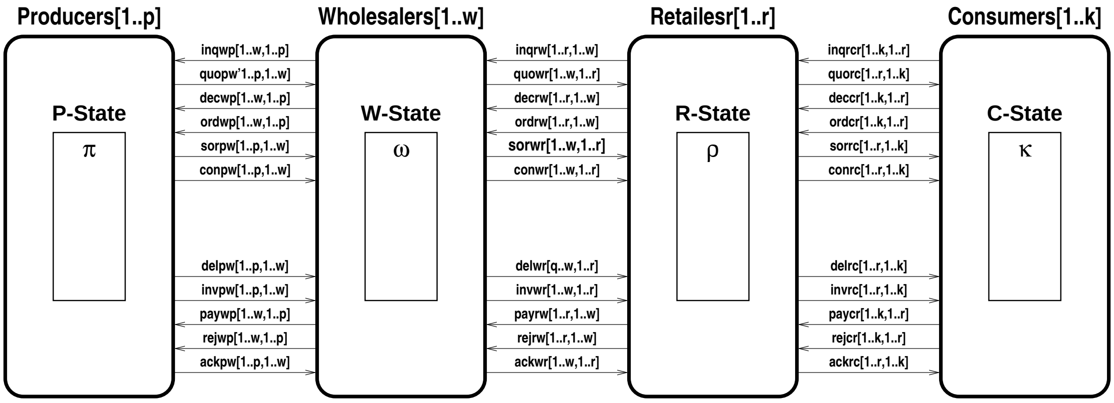
Section 7 briefly relate our somewhat detailed treatment of the conventional market to possible treatments of 'The E–Market'.

Section 8.1 finally discusses an essence of the previous sections (Sections 2–7).

2 Narrative: A Conventional Market System

We recall Figure 1 on page 4, redrawn as Figure 3:

Figure 3: The Conventional Market



2.1 The Behaviours

The market consists a number of "players": k parallel consumer *behaviours* concurrent with r parallel retailer *behaviours* concurrent with w parallel wholesaler *behaviours* concurrent with p parallel producer *behaviours*.[4]

These *behaviours* (ie. 'processes') interact by sharing in *events* such as queries, quotes, orders, confirmations, deliveries, etc. The *shared events* amounts to the communication of *information* between "players".[5]

The "player" *behaviours* each depend on local *information*.[6] Each "player" *behaviour* is understood as sequences of possibly concurrent *actions* and *events*. *Actions* (or as they may also be referred to: Tasks) are either generated spontaneously by the *behaviour* or are "triggered" by *external stimuli: Events*.

2.2 Transactions

Transactions are: *Queries, Quotes, Orders, Confirmations, Deliveries, Invoices, Accepts/Rejections, Payments,* and *Acknowledgements*.

- Each of the $k + r + w + p$ "players" can interact (in the formal model: synchronise and communicate), pairwise, with "neighbouring players": consumers with retailers, retailers with consumers and wholesalers, wholesalers with retailers and producers, and producers with wholesalers.

[4]The terms 'parallel' and 'concurrent' are here used synonymously.
[5]Events will be modelled by CSP–like output/inputs.
[6]Local information will be modelled in terms of states: Summaries of past events and actions.

- Interaction (ie. synchronisation and communication) is either tangible (ie. physical) or intangible (electronic). A consumer may turn up, physically, at the retailer, may (intangibly) inspect an item of merchandise on a store shelf, may (tangibly) take an item of merchandise off the shelf (thereby intangibly accepting this item), and may (tangibly) pay at the store counter with cash or (intangibly) pay by electronic means (credit card or other); or the consumer may (intangibly) inspect merchandise through a (tangible) paper catalogue, or through an (basically: intangible) web page; etc. Some deliveries are only possible if physical (ie. tangible), others are possible in both physical (tangible) or electronic (intangible) form (viz.: information, in general).

  Pragmatically speaking, and irrespective of whether one "player" (the 'buyer') is a consumer, a retailer or a wholesaler, and the other "player" (the 'seller') is a retailer, a wholesaler, respectively a producer:

  - A *query* serves to establish whether certain quantities of certain merchandises could possibly be *delivered,* when (date), and at which prices.
  - A *quote* normally responds to a *query* with the desired information — which may be partly negative.
  - An *order* normally and typically follows up on a *query/confirm* pair, requesting certain quantities of merchandise, at certain prices, to be *delivered* at certain dates (times).
  - A *confirm*[ation] normally responds to an *order* and expresses that the requested merchandise will indeed, or "sorry cannot", be delivered.
  - A *delivery* normally follows up on a "positive" outcome *order/confirm* pair, by the actual transfer of the requested merchandise.
  - An *acceptance/rejection* expresses that normally received merchandise was, respectively could not be, accepted. A *rejection* may or may not return the delivered merchandise. An *acceptance/rejection* can *accept* some merchandise and *reject* (remaining, delivered) merchandise.
  - An *invoice* normally follows in response to *accepted deliveries.*
  - A *payment* normally follows in response to an *accepted delivery.*
  - Finally a *payment* is *acknowledged.*

  Note our careful use of the hedge 'normally': In the domain all is possible. Players may misdirect quotes, confirmations, deliveries and invoices; players may order other than what has been inquired about (and/or making erroneous assumptions about available merchandise, quantities and prices), etc.

2.3 Discussion

To understand essences of E–Commerce [Semantics], we claim, is to first understand 'The Conventional Market'.

We have here "boiled" that down to understanding the four market "players": Consumers, retailers, wholesalers and producers — the buyers and sellers.

Their understanding, as we shall now see, is a matter of understanding their behaviours: Amongst them and internally, in general (Section 3); within them, specifically (Section 4); and the actions and the information these actions require and generate (Section 5).

That is: We arrive at "Data" Semantics (Section 5) only after careful analysis of flow (Section 3) and action (Section 4) semantics.

3 A "Formal" Model of System and "Player" Behaviours

3.1 Overall System Behaviour

We 'rough sketch' a formal model (hence the double quote around the above term 'formal').

The formal model is expressed in RSL [8], the RAISE [9] Specification Language.

The context and state types and the overall system process:

```
type
   [ generic transaction "messages": ]
   M = Inquiry | Quotation | Order | Confirm | Decline | Delivery |
          Accept | Reject | Invoice | Payment | Acknowledge | ...

   [ consumer, retailer, wholesaler and producer state names and state spaces: ]
   κ:kΣ, ρ:rΣ, ω:wΣ, π:pΣ

value
   [ fixed numbers of consumers, retailers, wholesaler and producers ]
   kid,rid,wid,pid: Nat

channel
   kr[ 1..kid,1..rid ] M /* later "expands" into many separate channels */
   rw[ 1..rid,1..wid ] M /* later "expands" into many separate channels */
   wp[ 1..wid,1..pid ] M /* later "expands" into many separate channels */

   /* `Expanded' channels: */
   /* inqxy,quoyx,ordxy,conyx,decxy,delyx,invyx,accxy,rejxy,payxy,ackyx */
   /* where x=con/y=ret, or x=ret/y=who, x=who/y=pro */

type
   ConId = {1..kid} /* Index set of Consumers */
   RetId = {1..rid} /* Index set of Retailers */
   WhoId = {1..wid} /* Index set of Wholesaler */
   ProId = {1..pid} /* Index set of Producers */

   kmΣ = ConId →m kΣ [ indexed consumer state spaces ]
   rmΣ = RetId →m rΣ [ indexed retailer state spaces ]
   wmΣ = WhoId →m wΣ [ indexed wholesaler state spaces ]
   pmΣ = ProId →m pΣ [ indexed producer state spaces ]

value
   kmσ:kmΣ [ specific, initial consumers' state space ]
   rmσ:rmΣ [ specific, initial retailers' state space ]
   wmσ:wmΣ [ specific, initial wholesalers' state space ]
   pmσ:pmΣ [ specific, initial producers' state space ]

   system: → Unit
   system() ≡
      || { consumer(k,kmσ(k)) | k:ConId } ||
      || { retailer(r,rmσ(r)) | r:RetId } ||
      || { wholesaler(w,wmσ(w)) | w:WhoId } ||
      || { producer(p,pmσ(p)) | p:ProId }
```

Annotations:

- $m : M$: We assume, initially, in the domain, that queries of consumers, retailers and wholesalers can be uniformly treated. Likewise for all other 'commands'.
- The "player" process definitions — recall, from above, that kid, rid, wid, pid are globally known constants (designating number of consumers, retailers, wholesaler and producers, respectively).
- xσ:xmσ(i): Each "player" has an own state space ($x\Sigma$, x: k, r, w, or p). State instances will be referred to by possibly primed κ, ρ, ω respectively π.
- Each "player" has (within its class) a unique index (k, r, w, p).
- The channel declarations: kr[1..kid,1..rid], rw[1..rid,1..wid], and wp[1..wid,1..pid], are only indicative. As we shall see, kr[k,r] will, instead of being one channel between a consumer (k) and a retailer (r), will become eleven channels; but see below.

3.2 " Formal, Player" Processes

In Section 4 we shall detail a specific version of a generic consumer behaviour and the corresponding (the consumer–interfacing) part of a generic retailer behaviour: Two that behave "nicely", as one would expect from some requirements specification — perhaps: Orderly sequences of transactions.

In reality each pair of neighbouring "players" [consumers: (consumer,retailer); retailers: (retailer,consumer) and (retailer,wholesaler); wholesalers: (wholesaler,retailer) and (wholesaler,producer); and producers: (producer,wholesaler)] — each pairing — at any one moment may engage in an interaction [in a transaction, in a shared event].

In the domain, ie. in the reality which we are modelling, these transactions may be generated and hence received in any order, "helter–skelter": There is no guarantee, in the domain, that the "players" behave orderly: Deliveries may be made without any prior order confirmations, let alone orders; invoices may be sent without any deliveries; etc. Some "players" behave professionally, some sloppily, some delinquently, and a few even outright criminally. And: To design proper systems we must capture, in the domain description, all such human behaviours.

To model this we must allow each "player" to:

- non-deterministically internally choose ($\sqcap$) which transaction it wishes to generate [this is modelled in basically the line of the function definitions below starting with "$\sqcap$ { **let**"];
- or to non-deterministically internally choose ($\sqcap$) whether, instead of generating such a transation, it instead wishes to declare willingness to engage in transactions generated by some neighbouring "player" [this is modelled in basically the line of the function definitions below starting with "$\sqcap$ ($\square$ { **let** ...)"];
- or to non-deterministically **skip** doing anything altogether !

3.2.1 The Generic Consumer Behaviour

value
  consumer: k:ConId × kΣ → **out**,**in** kr[k,1..rid] **Unit**
  consumer(k,κ) ≡
    let κ'' =
      $\sqcap$ { **let** ((m,r),κ') = k_Con_Ret_Fct_i(κ) **in** kr[k,r] ! m ; κ' **end** | i:CR_FIdx }
      $\sqcap$ ($\square$ { **let** m = kr[k,r] ? **in** k_Ret_Con_Fct(m,r)(κ) **end** | r:RetId })
      $\sqcap$ (**skip** ; κ)
    in consumer(k,κ'') **end**

  k_Con_Ret_Fct_i: k_Con_Ret_Fct
  k_Ret_Con_Fct: M × RetId → kΣ → kΣ

type
  i:CR_FIdx /* Index of consumer−to−retailer functions */
  k_Con_Ret_Fct = kΣ → (M × RetId) × kΣ

Annotations:

- Since **skip** ; b ≡ b for any b we shall omit **skip** ; in future.
- The k_Con_Ret_Fct_i functions shall model any of the inquiry, order, accept, reject, or payment transactions as generated by a consumer,
- and the k_Ret_Con_Fct function shall model the handling of received quote, confirm, delivery, invoive and acknowledgement transactions as generated by a (paired) retailer.
- k_Con_Ret_Fct_i is an indexed set of functions, and the non-determinism ranges over the index: Which of inquiry, order, accept, reject, or payment transactions is chosen can not be assumed. But, in a well-behaved world, k_Con_Ret_Fct_i inspects the state to decide an appropriate next transaction — except of perhaps inquiry or order.
- k_Ret_Con_Fct is a determinate function: It decides from its argument (m) which recording to make in the state — to be possibly later inspected by some k_Con_Ret_Fct_i (to decide its action).

The above "scheme" applies, inter alia, to the description of retailer, wholesaler and producer behaviours.[7]

3.2.2 The Generic Retailer Behaviour

value
  retailer: r:RetId × rΣ → **out**,**in** kr[r,_],rw[r,_] **Unit**
  retailer(r,ρ) ≡ **let** ρ′ = ret2con(r,ρ) ⌈⌉ ret2who(r,ρ) ⌈⌉ ρ **in** retailer(r,ρ′) **end**

  ret2con: r:RetId × rΣ → **out**,**in** kr[r,_] rΣ
  ret2con(r,ρ) ≡
      ⌈⌉ { **let** ((m,k),ρ′) = r_Ret_Con_Fct_i(ρ) **in** kr[k,r] ! m ; ρ′ **end** | i:RC_FIdx }
      ⌈⌉ ([] { **let** m = kr[k,r] ? **in** r_Con_Ret_Fct(m,k)(ρ) **end** | k:ConId })

  ret2who: r:RetId × rΣ → **out**,**in** rw[r,_] rΣ
  ret2who(r,ρ) ≡
      ⌈⌉ { **let** ((m,w),ρ′) = r_Ret_Who_Fct_i(ρ) **in** rw[r,w] ! m ; ρ′ **end** | i:RW_FIdx }
      ⌈⌉ ([] { **let** m = rw[r,w] ? **in** r_Who_Ret_Fct(m,w)(ρ) **end** | w:WhoId })

  r_Ret_Con_Fct_i: r_Ret_Con_Fct
  r_Con_Ret_Fct: M × ConId → rΣ → rΣ
  r_Ret_Who_Fct_i: r_Ret_Who_Fct
  r_Who_Ret_Fct: M × WhoId → rΣ → rΣ
type
  i:RC_FIdx /* Index of retailer−to−consumer functions */
  i:RW_FIdx /* Index of retailer−to−wholesaler functions */
  r_Ret_Con_Fct = rΣ → (M × RetId) × rΣ
  r_Ret_Who_Fct = rΣ → (M × WhoId) × rΣ

[7] We could, in a possibly submotted version of this report, omit these definitions, or refer them to an appendix ?

3.2.3 The Generic Wholesaler Behaviour

```
value
   wholesaler: w:WhoId × wΣ → out,in rw[_,w],wp[w,_] Unit
   wholesaler(w,ω) ≡ let ω′ = who2ret(w,ω) ⊓ who2pro(w,ω) ⊓ ω in wholesaler(w,ω′) end

   who2ret: w:WhoId × wΣ → out,in rw[_,w] wΣ
   who2ret(w,ω) ≡
         ⊓ { let ((m,r),ω′) = w_Who_Ret_Fct_i(ω) in rw[r,w] ! m ; ω′ end | i:WR_FIdx }
         ⊓ ( [] { let m = rw[r,w] ? in w_Ret_Who_Fct(m,r)(ω) end | r:RetId } )

   who2pro: w:WhoId × wΣ → out,in wp[w,_] wΣ
   who2pro(w,ω) ≡
         ⊓ { let ((m,p),ω′) = w_Who_Pro_Fct_i(ω) in  wp[w,p] ! m ; ω′ end | i:WP_FIdx }
         ⊓ ( [] { let m = wp[w,p] ? in w_Pro_Who_Fct(m,p)(ω) end | p:ProId } )

   w_Who_Ret_Fct_i: w_Who_Ret_Fct
   w_Ret_Who_Fct: M × RetId → rΣ → rΣ
   w_Who_Pro_Fct_i: w_Who_Pro_Fct
   w_Pro_Who_Fct: M × ProId → rΣ → rΣ
type
   i:WR_FIdx /∗ Index of wholesaler−to−retailer functions ∗/
   i:WP_FIdx /∗ Index of wholesaler−to−producer functions ∗/
   w_Who_Ret_Fct = wΣ → (M × RetId) × wΣ
   w_Who_Pro_Fct = wΣ → (M × ProId) × wΣ
```

3.2.4 The Generic Producer Behaviour

```
value
   producer: p:ProId × pΣ → out,in pq[p,1..wid] Unit
   producer(p,π) ≡
      let π″ =
         ⊓ { let ((m,w),π′) = p_Pro_Who_Fct_i(π) in wp[w,p] ! m ; π′ end | i:P_O_FIdx }
         ⊓ ( [] { let m = wp[w,p] ? in p_Who_Pro_Fct(m,w)(π) end | w:WhoId } )
         ⊓ π
      in producer(p,π″) end

   p_Pro_Who_Fct_i:p_Pro_Who_Fct
   p_Who_Pro_Fct: M × RetId → pΣ → pΣ
type
   p_Pro_Who_Fct = pΣ → (M × RetId) × pΣ
```

3.3 Discussion

We have modelled the domain behaviour of market "players".

Some comments are in order:

- **Parameterisation:** We could have "lifted" the process models of the buying facets of consumers, retailers and wholesalers into one parameterised behaviour description:

```
type
   Parameters == Buyer | Seller
   Buyer == Consumer | Retailer | Wholesaler
```

Seller == Retailer | Wholesaler | Producer
...
value
buyer: Buyer × ... → ... **Unit**
seller: Seller × ... → ... **Unit**

such that the result of instantiating, for example the parameterisable buyer behaviour, with mkBuyer(mkWholesaler(...)) yields the who2pro behaviour.

One can indeed claim that an essence of E–Commerce [Semantics] is the uniform treatment of buyers and sellers, whether consumers, retailers or wholesaler, respectively whether retailers, wholesalers or producers.

Parameterisation brings that uniformity out to its "fullest".

- We have only modelled the "loosest" aspects of these behaviours: Not any orderly behaviour.

An essence of E–Commerce [Behaviour Semantics] is to see to it that only desirable, orderly behaviours occur.

To the modelling of such orderly behaviours we turn next.

The aim of the present subsection has therefore been to highlight some aspects of what that might entail.

4 An "Orderly" Consumer ↔ Retailer Behaviour

In this section we shall define — almost as if it were a requirements — how a "pairing" of a consumer with any retailer might behave, when behaving in an orderly fashion. We model an entire conditional sequence from inquiry via quote, order, confirm, delivery, invoicing, acceptance, rejection, payment to acknowledgement.

4.1 A Narrative on a Transaction Protocol

We recall:[8]

- A consumer inquires as to availability (and, if available, price and delivery terms) of designated merchandise;
- the inquired retailer quotes — either positively or not so;
- the (same, or another) consumer orders some merchandise (perhaps stating previously obtained quotes);
- the quoting retailer confirms order;
- the retailer delivers ordered merchandise,
- and invoices the consumer;
- the consumer either accepts some or all of the delivered merchandise and/or
- rejects some or all, and returns that/those to retailer;
- finally the consumer pays accepted goods;
- and the retailer acknowledges payment.

[8] The itemized list may be omitted.

4.2 Formal Model of States and Transactions

type
[1] κ:kΣ,
    ρ:rΣ,
[2] conid:ConId = **Nat**,
    retid:RetId = **Nat**

[3] Trans == Inquiry | Quote | Order | Confirmation_Or_Sorry | Confirmation |
        Sorry | Delivery | Invoice | Accept_Or_Reject | Payment | Rejection | Declination
    ic:Inquiry, qr:Quote, oc:Order, csr:Confirmation_Or_Sorry,
    cr:Confirmation, sr:Sorry, dr:Delivery, ir:Invoice,
    accrej:Accept_Or_Reject, pc:Payment, rc:Rejection, dc:Declination
[4] Three_Way == confirmation_acceptable | confirmation_not_acceptable | sorry_no_confirmation
[5] ord_or_dec:OrdOrDec = order | declination

Annotations:

1 Defines state name prefixes and consumer and retailer state sorts.

2 Defines consumer and retailer index name prefixes and index types as natural numbers.

3 Defines value name prefixes and respective sorts for a number of consumer to retailer (suffix c) and retailer to consumer (suffix r) communications (whether physical, ie. tangible, or intangible).

4 Either a retailers' confirmation is acceptable to consumer, or it is not, or retailer cannot confirm order.

5 A retailer confirmation either leads to a consumer order or a consumer declination.

4.3 Formal Model of Means of Interaction

We define the various consumer to retailer, suffix cr, and retailer to consumer, suffix rc communication "paths", ie. events. Channel name prefixes correspond to first three characters of the type (ie. sort) name of the communicated values ("messages").

channel
  inqcr[1..cid,1..rid] Inquiry
  quorc[1..rid,1..cid] Quote
  ordcr[1..cid,1..rid] Order
  conrc[1..rid,1..cid] Confirm
  sorrc[1..rid,1..cid] Sorry
  delrc[1..cid,1..rid] Delivery
  invrc[1..rid,1..cid] Invoice
  paycr[1..cid,1..rid] Payment
  rejcr[1..cid,1..rid] Rejection
  deccr[1..cid,1..rid] Decline

An essence of E–Commerce [Interaction Semantics] is that of identifying — and fully automating or partially supporting — all conceivable "lines of communication" between (all) "players".

4.4 Local Consumer and Retailer Function Signatures

Next we first define functions 'local' to the consumer — by just stating their signature:

```
value
  Mk_Inquiry: kΣ × ConId → Inquiry × kΣ
  obs_Retailer_Id: Inquiry → RetId
  is_Quote_OK: Quote → kΣ → Bool
  Mk_Order: Inquiry × Quote → kΣ → Order × kΣ
  Confirm_or_Sorry: (Confirm|Sorry) → kΣ → Three_Way × kΣ
  is_Confirm_OK: Confirm → kΣ → Bool
  Remember_Confirm: Inquiry × Quote × Confirm → kΣ → kΣ
  Accept_or_Reject: Delivery → kΣ → AccOrRej × kΣ
  is_Accept: AccOrRej → kΣ → Bool
  Mk_Payment: Invoice → kΣ → Payment × kΣ
  Mk_Rejection: Delivery → kΣ → Rejection × kΣ
  Mk_Decline: (Quote|Confirm) → kΣ → Decline × kΣ
```

Then we define functions 'local' to the retailer — again by just stating their signature:

```
value
  Mk_Quote: Inquiry → rΣ → Quote × rΣ
  is_Order_or_Declination: (Order|Decline) → rΣ → OrdOrDec
  Mk_Confirmation_or_Sorry: Inquiry×Quote×Order → rΣ → ConfirmationOrSorry × rΣ
  Mk_Confirmation: ConfirmationOrSorry → rΣ → Confirmation × rΣ
  Mk_Delivery: Inquiry×Quote×Order×Confirmation → rΣ → Delivery × rΣ
  Mk_Invoice: Inquiry×Quote×Order×Confirmation → rΣ → Invoice × rΣ
  Record_Payment_or_Rejection: (Payment|Rejection) → rΣ → rΣ
  Sorry_out_of_Stock: ConfirmationOrSorry → rΣ → Sorry × rΣ
  Rcord_Declination: Declination → rΣ → rΣ
```

An essence of E–Commerce [Action Semantics] is that a great number of local functions be either fully automated or partially supported.

4.5 An Orderly Consumer Behaving

We present a "partial" definition of an orderly consumer behaviour: That part which "interfaces" to any — similarly orderly — retailer behaviour.

```
value
  consumer: k:ConId × rΣ →
     out inqcr[k,_],ordcr[k,_],paycr[k,_],rejcr[k,_],deccr[k,_]
     in quorc[_,k],conrc[_,k],sorrc[_,k],delrc[_,k],invrc[_,k] Unit
  consumer(k,κ) ≡ let κ′ = con2ret(k,κ) in consumer(k,κ′) end

  con2ret: k:ConId × rΣ →
     out inqcr[k,_],ordcr[k,_],paycr[k,_],rejcr[k,_],deccr[k,_]
     in quorc[_,k],conrc[_,k],sorrc[_,k],delrc[_,k],invrc[_,k] rΣ
  con2ret(k,κ0) ≡
     let (ic,κ1) = Mk_Inquiry(κ0,k),
         r = obs_Retailer_Id(ic) in
     inqcr[k,r] ! ic ; ...
     let qr = quorc[r,k] ? in
     let (quote_ok,κ2) = Is_Quote_OK(qr)(κ1) in
     if quote_ok
```

```
        then
          let (oc,κ3) = Mk_Order(ic,qr)(κ2)
          in ordcr[k,r] ! oc
          let cr_or_sr = conrc[r,k] ? ⌷ sorrc[r,k] ? in
          let (ok_or_not,κ4) = Confirm_or_Sorry(cr_or_sr)(κ3) in
          cases ok_or_not of
            confirmation_acceptable →
              let κ5 = Remember_Confirm(ic,qr,cr)(κ4) in
              let (dr,ir) = (delrc[r,k] ?, invrc[r,k] ?) in
              let (accrej,κ6) = Accept_or_Reject(dr)(κ5) in
              let κ9 =
                if is_Accept(accrej)(κ4)
                  then
                    let (pay,κ7) = Mk_Payment(ir)(κ6) in
                    paycr[k,r] ! pay ;
                    κ7 end
                  else
                    let (rej,κ8) = Mk_Rejection(dr)(κ6) in
                    rejcr[k,r] ! rej ;
                    κ8 end
                end in
              let ack = ackrc[k,r] ? in κ9 end ;
              end end end,
            confirmation_not_acceptable →
              let (dec,κ10) = Mk_Decline(qr)(κ4) in
              deccr[k,r] ! dec ;
              κ10 end
            sorry_no_confirmation → κ4 end
          end end end
        else
          let (dec,κ11) = Mk_Decline(qr)(κ2) in
          deccr[k,r] ! dec ;
          κ11
      end end end end end
```

4.6 An Orderly Retailer to Consumer Behaviour

We present a "partial" definition of an orderly retailer behaviour: That part which "interfaces" to any — similarly orderly — consumer behaviour.

Observe the "pairing" of outputs (!) from the consumer with inputs (?) to the retailer, and, vice-versa, the outputs from the retailer with inputs to the consumer.

```
value
  ret2con: r:RetId × rΣ →
     in inqcr[_,r],ordcr[_,r],paycr[_,r],rejcr[_,r],deccr
     out quorc[r,_],conrc[r,_],sorrc[r,_],delrc[r,_],invrc[r,_] kΣ
  ret2con(r,ρ0) ≡
     ⌷ {
      let ic = inqcr[k,r] ? in
      let (qr,ρ1) = Mk_Quote(ic)(ρ0) in
      quorc[r,k] ! qr ; ...
      let oc_or_dc = ordcr[k,r] ? ⌷ deccr[k,r] ? in
      let (ord_or_dec,ρ2) = is_Order_or_Declination(oc_or_dc)(ρ1) in
```

```
    cases ord_or_dec of
      order →
        let (csr,ρ3) = Mk_Confirmation_or_Sorry(ic,qr,oc_or_dc)(ρ2) in
        if isConfirmation(cr)(ρ3)
          then
            let (cr,ρ4) = Mk_Confirmation(csr)(ρ3) in
            conrc[r,k] ! cr ; ... ;
            let (dr,ρ5) = Mk_Delivery(ic,qr,oc_or_dc,cr)(ρ4) in
            let (ir,ρ6) = Mk_Invoice(ic,qr,oc_or_dc,cr)(ρ5) in
            (delrc[r,k]!dr,invrc[r,k]!ir) ;
            let prc = paycr[k,r]?⌷rejcr[k,r]? in
            let (ρ7,ack) = Record_Payment_or_Rejection(prc)(ρ6) in
            ackrc[r,k] ! ack ;
            ρ7 end end end end end
          else
            let (sr,ρ8) = Sorry_out_of_Stock(csr)(ρ3) in
            sorrc[r,k] ! sr ;
            ρ8 end
      declination →
        let ρ7 = Record_Declination(oc_or_dc)(ρ2) in
        ρ7 end
    end end end end end end end
  | k:ConId }
```

4.7 Discussion

An essence of E–Commerce [Action Semantics] is to see to it that only desirable, orderly protocols of behaviours occur.

We have just illustrated one. Many protocols are desirable.

Any one "player", buyer or seller, may wish to alternate (to "multiplex") between transacting with one "player", for a while, and then "shifting attention" to another "player", for a while. Each of these "whiles" may entail one or more transactions — a inquiry, or an inquiry and a quote, or a delivery and invoicing followed by acceptance and payment, etc.

The "player" behaviours, outlined in Sections 3.2.1–3.2.4, allow for this multiplexing.

The formalisations of specific, "order constrained" action behaviours, of the present sections' Subsections 4.5–4.6, need therefore be "chopped up" into the kind of index function parts illustrated in for example Sections 3.2.1–3.2.2.

5 Information Analysis — from Sorts to Concrete Types

5.1 "Player" Contexts and States — Informally

Items within a unique class of merchandise have a common unique merchandise identifier. Retailer contexts contain a catalog, and retailer states contain a store and consumer and wholesaler transaction databases. These latter record all transactions performed with consumers, respectively wholesalers. Stores record quantity-on-hand for each merchandise. If quantity-on-hand is zero for some merchandise, then the catalog records delivery time. The retailer catalog also records merchandise prices. Here we assume a simple, single unit price. Wholesaler catalogs may record prices depending on either quantity or delivery time. One can imagine prices that depend on both, and/or on consumer (ie. retailer) loyalty, etc.

MnQ thus captures current quantities_on_hand. The various MnQ, MnQD, and MnQPD are intended to capture varieties of buyer inquiries: A certain quantity of merchandise; a certain quantity of merchandise at or before a certain date; a certain quantity of merchandise at or below a certain price; and a certain quantity merchandise at or before a certain date and at or below a certain price.

Uid stand for a class of globally unique identifiers that are normally first issued when a first transaction of an expected sequence of one or more such transactions are commenced. Usually the identifier can be thought of as concocted from the identifiers of the producer, wholesaler, retailer, and consumer. Usually consumers do not issue such identifiers. So a retailer will affix a consumers' first (inquiry) transaction with an appropriate (consumer id, retailer id and data-time stamp concocted) identifier. But usually transaction initiating retailers, wholesalers and producers can supply initial transaction identifiers.

5.2 "Player" Contexts and States — Formally

Usually transactions are carried out in a *configuration* of a spectrum from *context* to *state* values. By a *context* component we understand a component whose value is static, ie. which is a constant: Never changes. By a *state* component we understand a component whose value is dynamic, temporal, ie. which varies: Often changes. Some component values change rarely, some not so often.

Retailer stores and wholesaler and producer warehouses, as well as information bases, represent states; their catalogues contexts; producer production plans "fall" somewhere in-between.

type
  Mn ≡ Merchandise_name
  Q ≡ Quantity, Q = **Nat**
  P ≡ Price, P = **Nat**
  D ≡ Delivery_time
  DateTime

  Con_IB = (DateTime $\overrightarrow{m}$ Trans) ...
  Ret_IB = (DateTime $\overrightarrow{m}$ Trans) ...
  Who_IB
  Pro_IB

  Σ ≡ State ≡ Store ≡ Warehouse = MnQ

  MnQ = Mn $\overrightarrow{m}$ Q
  MnQD = Mn $\overrightarrow{m}$ (Q × D)
  MnQPD = Mn $\overrightarrow{m}$ (Q × P × D)

  kΣ = Con_IB

  rΣ = Σ × R_CTL × Ret_IB
  R_CTL = Mn $\overrightarrow{m}$ (P × D)

  wΣ = Σ × W_CTLs × Who_IB
  W_CTLs = Mn $\overrightarrow{m}$ (DPT × QPT)
  DPT = D $\overrightarrow{m}$ (Q × P)
  QPT = Q $\overrightarrow{m}$ (P × D)

  pΣ = Σ × P_CTL × PPP × Pro_IB
  PPP = Mn $\overrightarrow{m}$ (D $\overrightarrow{m}$ (Q $\overrightarrow{m}$ P)) /* Production Plan */
  P_CTL

  Uid
value

obs_Ids: Uid → (ProId|WhoId|RetId|ConId)-**set**

An essence of E–Commerce [Configuration Semantics] is to carefully identify the spectrum of from static (context) to dynamic (state) components — and to allow remote access to and possible updates of these.

The above formalisation does not claim to be fully representative of what is needed. But it shows what it takes.

5.3 Transactions — Informally and Formally

Transaction kinds can be "gleaned" from the previous descriptions — as captured most succinctly in the consumer and retailer process definitions.

Trans = Inquiry | Quote | Order | Confirmation_Or_Sorry | Confirmation | Sorry |
Delivery | Acceptance_Or_Rejection | Acceptance | Rejection | Payment | Declination

The particular concrete content of each transaction follows after some analysis. The reader is expected to use a couple of fingers on the pairs of consumer and retailer process definitions. Pairs of fingers are placed at/and/around where pairs of synchronising communications (ie. events) are prescribed. This in order that the reader may either validate our choice of concrete type, or while also, ie. co–, discovering a suitable such concrete type for each transaction.

We illustrate but a few cases:

- Inquiry: Consumers are asumed to initiate inquiries "out of the blue": Maybe some marketing campaign has initiated some conditions of the consumer state.

  In any case we assume only that the consumer directs the inquiry at a specific retailer (RetId) and asks for prices of quantity and possibly dates of delivery (MnQD).

  In order to "fit" inquieries into their related quotes, and these into orders, confirmations (or "sorry, we meanwhile ran out of stock"), deliveries, acceptances, rejections, invoices, etc., we decide to structure the inquiry information "similarly" to the information given in those other transactions.

- Quote: Quotations augments inquiries with dates of delivery, if deliverable — otherwise we can think of the quote only listing that merchandise which can be delivered, or list undeliverable merchandise by either zero quantities or infinite dates: Remember this is "only" a specification of abstract transactions, not ther concrete, "user–friendly" and displayed rendition.

  An inquiry — which "triggers" the quotes we are now describig — can be thought of as a first transaction, hence the seller (retailer, or wholesaler, or producer) will associate it with a unique identifier (Uid) and the buyer (consumer, retailer, resp. wholesaler) will be informed of this identification (for future reference).

- Order: The buyer unconditionally orders certain quantities of certain merchandise, no matter (for example) price and delivery dates, and conditionally orders other other merchandise at, or below, buyer stated prices and by, or before buyer stated dates (OCSDI).

- Confirmation_Or_Sorry: The seller confirms which parts can be delivered, and which parts it, "sorry" cannot deliver.

  The above is a deviaton from our example definition of (in this case the) retailer response, see the **if** isConfirmation(cr)(ρ3) line on Page 19.

  We do not deviate out of sloppiness, but more to indicate that we are only documenting a stage in a domain specification stage. Which choice is taken, whether to combine confirmation with "sorry", as above, or separate them, as shown on Page 19 and indicated below, is a matter for discussion with real customers.

- Confirmation: Here we assume, in contrast to immediately above, that an entire order can be somehow delivered. The confirmation states price and delivery dates.
- Since we wish to treat transaction uniformly, we equip each with a pair of "player" identifiers: That of the buyer, and that of the seller.
- *&c.*

5.4 Formal Transaction Types

BuyId = ConId | RetId | WhoId
SellId = RetId | WhoId | ProId
BS_Id == ids:(bid:BuyId sid:SellId)

InqCmd = idp:BS_Id inq:MnQ
GenCmd = idp:BS_Id ms:MnQPD uid:Uid
Ord_Field = un_cond:MnQ cond:MnQPD
SimCmd = idp:BS_Id MnQ uid:Uid

Inquiry == InqCmd
Quote == GenCmd
Order == ids:BS_Id Ord_Field uid:Uid
Confirmation_Or_Sorry == (Confirmation|Sorry) uid:Uid
Confirmation == GenCmd
Sorry == SimCmd
Delivery == GenCmd
Invoice == GenCmd
Acceptance_Or_Rejection == acc:SimCmd rej:SimCmd
Acceptance == SimCmd
Rejection == SimCmd
Payment == GenCmd
Declination == SimCmd
Acknowledgement == GenCmd

Each transaction usually follows a transaction preparation as performed by some consumer or retailer local function.

An essence of E–Commerce [Transaction Semantics] is to identify all possible transactions, to structure them in classes (not shown above), and to either fully automate or partially support their generation.

5.5 Consumer Transaction Function Definitions

First we discuss one consumer local functions:

value
Mk_Inquiry: ConId $\rightarrow$ kΣ $\rightarrow$ Inquiry $\times$ kΣ
Mk_Inquiry(k)(κ) $\equiv$
let mns:Mn-**set** • mns $\in$ obs_Mns(κ),
r:RetId • r $\in$ obs_RetIds(κ) **in**
let mnq = [mn $\mapsto$ q | mn:Mn,q:Q • mn $\in$ mns] **in**
let inquiry = mkInquiry(k,r,mnq) **in**
let κ' = Record_in_Con_IB(inquiry)(κ) **in**
(inquiry,κ') **end end end end**

obs_Retailer_Id: Inquiry $\rightarrow$ RetId obs_Retailer_Id(mkInquiry(k,r,mnq)) ≡ r

Record_in_Con_IB: Inquiry → kΣ → kΣ
Record_in_Con_IB(mkInquiry(k,r,mnq))(κ) **as** κ′
post ∃ dt:DateTime • dt ∈ **dom** κ′ ∧ mkInquiry(k,r,mnq) = κ′(dt)

The definition expresses that the consumer arbitrarily, ie. non–deterministically selects merchandise, their quantities and a retailer — both as a result of inspecting the consumer "information base". Then the user notes, in that same consumer "information base", that such a query has been "concocted".

An essence of E-Commerce [Information Semantics], shown by this example, is that we learn more about what the consumer "information base" may be, through which observer functions are needed. Whether, as in the domain, this "information base" is just the consumer's memory, or is in the form of some paper catalogues, or is the result "gleaning" the internet, need not be detailed. Not in the domain. But it says something about what requirements we may wish for an actual consumer data base.

We list remaining consumer local function signatures:

value
is_Quote_OK: Quote → kΣ → **Bool**
Mk_Order: Inquiry × Quote → kΣ → Order × kΣ
Confirm_or_Sorry: (Confirm|Sorry) → kΣ → Three_Way × kΣ
is_Confirm_OK: Confirm → kΣ → **Bool**
Remember_Confirm: Inquiry × Quote × Confirm → kΣ → kΣ
Accept_or_Reject: Delivery → kΣ → AccOrRej × kΣ
is_Accept: AccOrRej → kΣ → **Bool**
Mk_Payment: Invoice → kΣ → Payment × kΣ
Mk_Rejection: Delivery → kΣ → Rejection × kΣ
Mk_Decline: (Quote|Confirm) → kΣ → Decline × kΣ

5.6 Retailer Transaction Function Definitions

First we discuss one retailer local function:

value
Mk_Quote: Inquiry → rΣ → Quote × rΣ
Mk_Quote(mkInquiry(k,r,mkMnQ(mnq)))(κ) ≡
let (uid,κ′) = Get_Uid(κ) **in**
(mnqpd,κ″) = Try_to_Match_Inquiry(mnq)(κ′) **in**
let quote = mkQuote(k,r,mnqpd,uid),
let κ‴ = Record_in_Ret_IB(quote)(κ″) **in**
(quote,κ‴)
end end end

The above function relies on an auxiliary function.

value
Try_to_Match_Inquiry: MnQD → rΣ → Quote × rΣ
Try_to_Match_Inquiry(mnqd)(κ) **as** mnqpd
post ...

is_Match: MnQ × MnQPD → **Bool**
is_Match(mnq,mnqpd) ≡

$$
\begin{array}{l}
\quad \mathbf{dom}\ \mathrm{mnq} = \mathbf{dom}\ \mathrm{mnqpd} \wedge \\
\quad \forall\ \mathrm{m{:}Mn} \bullet \mathrm{n} \in \mathbf{dom}\ \mathrm{mnq} \Rightarrow \\
\qquad \mathbf{let}\ \mathrm{cq} = \mathrm{mnq(m)},\ \mathrm{(rq,rp,rd)} = \mathrm{mnqpd(m)}\ \mathbf{in} \\
\qquad \mathrm{rq} \leq \mathrm{cq} \wedge (\mathrm{rq} = 0 \wedge \mathrm{rd} \neq \mathrm{nil}) \wedge (\mathrm{r} \neq 0 \wedge \mathrm{rd} = \mathrm{nil})\ \mathbf{end}
\end{array}
$$

The retailer attempts to match the inquiry. Since we are in the domain — where all is possible, and not specifying requirements, where we constrain "all behaviours to desirable ones" — some that are not sloppy, not delinquent, and not criminal — since we are, to repeat, in the domain, we cannot specify too much, in fact, we can specify nothing !

If we decree that the quote must only mention merchandise queried by the consumer, then we can be "accused" of model a requirements – so we refrain from saying much.

But we can speak of such 'matches', in the domain, so we specify a predicate, is_Match. is_Match specifies that the two sets of merchandise names, that of the inquiry and that of the quotation, must be the same, and that if a deliverable quantity is (currently) zero, then a non-nil delivery date is given, etc.

Finally we list remaining retailer local function signatures:

value
is_Order_or_Declination: (Order|Decline) → rΣ → OrdOrDec
Mk_Confirmation_or_Sorry: Inquiry×Quote×Order → rΣ → ConfirmationOrSorry × rΣ
Mk_Confirmation: ConfirmationOrSorry → rΣ → Confirmation × rΣ
Mk_Delivery: Inquiry×Quote×Order×Confirmation → rΣ → Delivery × rΣ
Mk_Invoice: Inquiry×Quote×Order×Confirmation → rΣ → Invoice × rΣ
Record_Payment_or_Rejection: (Payment|Rejection) → rΣ → rΣ
Sorry_out_of_Stock: ConfirmationOrSorry → rΣ → Sorry × rΣ
Record_Declination: Declination → rΣ → rΣ

5.7 Discussion

An essence of E–Commerce [Information Semantics] derives from a careful analysis all the individual (ie. local) actions, the information they require in order to be successful, and the information they generate.

In this section (ie. Section–refdata-analysis) we have only "scratched the surface" of that issue. That is: We have shown "how to go about doing it" — in at least the way we would do so !

6 Example: An Unconstrained Market System

6.1 A Motivation

In Sections 3 and 3.2 we modelled the system of all "player" behaviours and the indivudual "player" behaviours. The models emphasize the arbitrariness, in the domain, of "player" behaviour.

To get a "feel" for what is going on we bring the below "snapshot".

The below model "abstracts" actual domain behaviours by showing only the possible events, ie. the possible synchronisations & communications between all four kinds of behaviours, that is: On the inter-behaviour events, not on the intra–, the internal behaviour control flows. Another way of formulating this is: The four process definitions next are to be thought of as concurrent behaviour specification with **if then else** specified control flow.

With respect to the detailed consumer and retailer process formalisation of Section 4 we have made a number of channel and function name simplifications or omissions: We have left out channel indices, we have left out most conditionals, and hence their functions, and we have simplified some remaining function names — all this to try fit all four process definitions, see Figure 4 on the facing page, side-by-side, onto one page.

6.2 The Formula "Snapshot"

Figure 4: A Schematic Description of an Unconstrained Market

```
con2ret() ≡
    ... ⊓ ...
    let ic=Inq() in
    inqcr!ic ;
    ... ⊓ ...
    let qr=quorc? in
    ... ⊓ ...
    let oc=Or(qr) in
    ordcr!ord ;
    ... ⊓ ...
    let cr=conrc? in
    ... ⊓ ...
    let dr=delrc?,
        ir=invrc? in
    ... ⊓ ...
    if Accept(dr)
    then paycr!Pay(ir)
    else rejcr!dr
    ... ⊓ ...
    con2ret()
    end end end
    end end end
```

```
retailer() ≡
    ... ⊓ ...
    let ic=inqcr? in
    ... ⊓ ...
    let qr=Quo(ic) in
    quorc!qr ;
    ... ⊓ ...
    let oc=ordcr? in
    ... ⊓ ...
    let cr=Con(oc) in
    conrc!cr ;
    ... ⊓ ...
    let dr=Del(cr),
        br=Inv(cr) in
    (delrc!dr
        ∥ invrc!br);
    ... ⊓ ...
    let pr=paycr?
        [] rejcr? in
    ... ⊓ ...
    let ir=Inq() in
    inqrw!ir ;
    ... ⊓ ...
    let qw=quowr? in
    ... ⊓ ...
    let or=Ord(qw) in
    ordrt!or ;
    ... ⊓ ...
    let cw=conwr? in
    ... ⊓ ...
    let dw = delwr?,
        iw = invwr? in
    ... ⊓ ...
    if Accept(dw)
    then pay!Pay(iw)
    else rejrw !dw
    ... ⊓ ...
    retailer()
    end end end end
    end end end end
    end end end end
```

```
wholesaler() ≡
    ... ⊓ ...
    let iw=Inq() in
    inqwp!iw ;
    ... ⊓ ...
    let qp=quopw? in
    ... ⊓ ...
    let ow=Ord(qp) in
    ordwp!ow ;
    ... ⊓ ...
    let c =conpw? in
    ... ⊓ ...
    let dp = delpw?,
        ip = invpw? in
    ... ⊓ ...
    if Accept(dp)
    then paywp!Pay(ip)
    else rejwp!dt
    ... ⊓ ...
    let ir=inqrw? in
    ... ⊓ ...
    let qw=Quo(ir) in
    quowr!qw ;
    ... ⊓ ...
    let or=ordcr? in
    ... ⊓ ...
    let cw=Con(or) in
    conwr!cw ;
    ... ⊓ ...
    let ip=invpw? in
    ... ⊓ ...
    let iw=Inv(ip),
            dw=Del(ip) in
    (invwr!iw
        ∥ delwr!dw);
    ... ⊓ ...
    let pr=payrw?
        [] rejrw? in
    ... ⊓ ...
    wholesaler()
    end end end end
    end end end end
    end end end end end
```

```
pro2who() ≡
    ... ⊓ ...
    let iw=inqwp? in
    ... ⊓ ...
    let qp=Quo(iw) in
    quopw!qp ;
    ... ⊓ ...
    let ow=ordwp? in
    ... ⊓ ...
    let cp=Ord(ow) in
    conpw!cp ;
    ... ⊓ ...
    let dp=Del(cp),
        ip=Inv(,cp) in
    (delpw!dp
        ∥ invpw!ip);
    ... ⊓ ...
    let pr=paywp?
        [] rejwp? in
    ... ⊓ ...
    pro2who()
    end end end
    end end end
```

Note, above, the internal non-deterministic choice (⊓) between the "two sides" of the retailer and the wholesaler behaviours: Those towards consumers and towards wholesalers, respectively those towards retailers and towards producers. Consumers and producers communicate only "towards one side", viz.: retailers, respectively wholesalers.

6.3 Discussion

An essence of E–Commerce [Information Semantics] is to make sure that all "players", buyers and sellers, can interact as freely as possible.

In this paper we have focused, somewhat, on that "freeness".

7 Towards Semantics of E–Commerce

7.1 Introduction

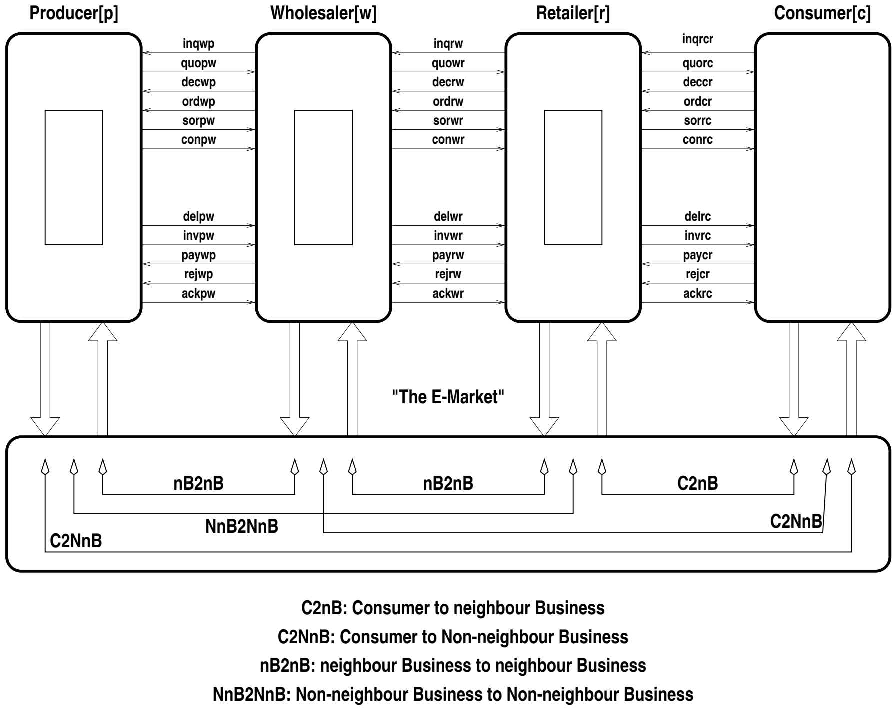
An essence of E–Commerce is that of the virtual market. One that gives the appearance of being (like) the physical, conventional market, only it is mostly, or all, "borne by electronics !".

To enable virtuality, it seems that a concept of 'Market Knowledge' is required. And to "carry" virtuality, it seems that a number of new "player" transactions are warranted. These "new" transactions occur between conventional buyers and sellers, on the one side, and the 'E–Market', on the other side,

We will just briefly cover these aspects — with the single purpose of showing that it can all be reduced to models like the ones given in Sections 3–5.

We recall the "The E–Market", Figure 2 on page 6, redrawn in Figure 5.

Figure 5: The E–Commerce Extended Market



In Figure 5 we show the possibility of either unspecified, ie. not specifically receiver targeted, or specifically such targeted, consumer–to–neighbouring–business, consumer–to–non-neighbouring- –business, neighbouring–business–to–neighbouring–business, non-neighbouring–business–to–non-neighbouring–business, 'transactions'.

7.1.1 Towards "Market Knowledge"

- In addition to the four "players" (consumers, retailers, wholesalers and producers) we now introduce a fifth "player": 'The E–Market'. 'The E–Market' can, idealising, be assumed to know the sum total of whatever any player can individually ascertain, ie. "find out" !
- Example: *Any consumer can, in turn, inquire with any retailer and obtain quotes. 'The E–Market' can therefore be assumed to yield similar, retailer–identified quotes. Similar*

for inquiries and quotes between retailers and wholesalers, and between wholesalers and producers.

- Now (consumer, retailer and wholesaler) inquiries can be directed at 'The E–Market', and 'The E–Market' will respond with sets of quotes, each member identified by the specific (retailer, wholesaler, respectively producer) source that will deliver as quoted.

For 'The E–Market' to be "all knowledgeable" we envisage that each of the retailer, wholesaler, and producer processes regularly communicate their own merchandise knowledge to 'The Market', namely whenever updated, ie. whenever changed. — ie. with 'Market' Inquiry and 'Market' knowledge (ie. awareness) Updates.

Likewise, but implicitly, all the "players", while transacting with other "players", via 'The E–Market', leave a "trail", a "trace" of their transactions.

So specifically do the consumers who wish to utilise that 'Market'. Their externally observable 'behaviours' (the inquieries, orders, etc.) may be recorded for statistical and other purposes.

```
type
   E_Trans = Knowledge_Update | ...
   Knowledge_Update == sid:Sid  mnqpd:MnQPD
   ...
```

• • •

The same principles and techniques as were applied in understand the conventional market now applies — inter alia — to develop descriptions of how 'The E–Market' may attain its knowledge.

7.1.2 Virtual Transactions

Some observations:

- Above we only dealt with 'Market' Updates. We now extend 'Market' functionalities.
- "Old" practices: Consumers only being delivered merchandise from retailers, retailers from wholesalers, etc., alll those "old" practices now are "under threat".
- An electronic commerce market could be one where any "player" may inquire any other "player", where deliveries may occur directly from producers to consumers, etc.
- Thus we may add a new (a sixth) set of players: A set of 'distributors' — while, of course, retaining the "old" system (of practices and habits).

• • •

The same principles and techniques as were applied in understand the conventional market now applies — inter alia — to develop descriptions of the new, 'The E–Market', transactions.

Specifically one (i) introduces channels between conventinal players and 'The Market', (ii) a 'Market' process (say market, etc. The market process is, at any time, willing to engage in transactions with any "player". In respnse to a *buyer inquiry* the market process responds with a *quote* that may list several sellers' offers, etc.

```
type
   E_Trans = ... | Buyer_Inquire | Buyer_Order | ...
   Buyer_Inquire == bid:BId  mnq:MnQ
   Buyer_Order == bid:Bid  sid:Sid  mnqpd:MnQPD
   ...
```


7.1.3 Discussion

We leave our rather cursory first treatment of "Towards an E–Commerce", to, instead, be guided by the issues raised in [7].

7.2 Relations to T.S.Dillon et al.

We briefly list some ways in which we would respond to issues raised by [7].

The *slash* font texts more-or-less derive from [7]. The sans serif texts constitute a response.

7.2.1 Emergent Business Model as Basis for E–Commerce Model ([7] Sect. 2.1)

[9]

Briefly: An existing business which embraces the traditional E–Commerce concept, cannot just leave its "old ways of doing things behind", give them up, abandon their use. That business (producer, wholesaler, retailer) must combine the two: The traditional way of doing business with the new, "electronic" and software programmed way. It is given only a few businesses, amazon.com *is a case in point, to only provide "clean" electronics + software commerce.*

Thus it is that [7] suggests the term E–Business to stand for a confluence of traditional business with electronics + software commerce into an 'Emergent Business' concept.

Response: The confluence referred to may, for example, take the following forms:

- Many of the 'output/input' (ie. the synchronisation & communication) events mentioned in the domain model be now interpreted specfically:
  - Either as only physical, hence tangible, events: Like the delivery of certain items of merchandise, or the physical presence of a buyer or seller.
  - Or as only electronic, hence intangible, events: Like the data communication conveyance of certain items of information: Queries, quotattions, orders, confirmations, invoices, payments, etc.
  - Or as both !
- Or it may be that certain pieces of information is both physically and electronically presentable.

Confluence emphasizes the last two possibilities, the 'both' positions.

We emphasize that the above reflects interpretations of what requirements express: Not only that the specified values (incl. behaviours) *"shall be"*, but also "how it shall (tangibly and/or intangibly) be".

7.2.2 Perspectives of E–Commerce ([7] Sect. 3.0)

According to [7] Kalakota and Winston [10] analyses four perspectives:

- **Communication:** *Electronic conveyance of information (typically by Internet).*

  Response: What was previously conveyed in tangible form — physically, eg. by sound (speaking, even if on the phone) and hearing, or in paper form (even when via fax): by physically writing and reading (seeing) — may now be requirements specified as being intangible: fully electronically communicable: Sent and received, ie. encoded and decoded.

  Before (E–Commerce) the communication was via human interpretation.

  With E–Commerce the communication is now "cutting out, short-circuiting" the human filtering: Some software is to "stand in" for certain, but not necessarily all, human processes.

- **Business Process:** *Automation of business transactions and workflow.*

[9] The numbering '2.1' in [7] seems wrong. Probably the section number should be: 2.0.

Response: The requirements specification of inquiry quotation–, order confirmation–, order delivery, invoicing and payment forming functions represents requirements specification of the *"Automation of business transactions"*, and the requirements specification of process interaction protocols represents requirements specification of the *"Automation of workflow"*.

- **Service:** *Customer service.*

  Response: Typically we here speak of extensions to the domain: Actions that might be spoken of, but never "implemented" in the domain, or "actions" which were simply infeasible. In an extended domain we can, for example, speak of searching the entire market (hence more "accurately") for all possible retail, wholesaler and/or producer sources of certain merchandise; or we can speak of retailers, wholesaler and/or producers "polling" the entire market (hence more "accurately") for possible targets for their merchandise.

  Figure 2 on page 6 hints at this. We shall later suggest such extended operations.

- **Online:** *Instant buying and selling.*

  Response: Specifying certain transaction sequences and that they follow a certain protocol (ie. a specific set of, albeit possibly alternative, behaviours), and given the fact that these sequences and protocols be computerised and conveyed by data communication (over the internet or otherwise), amounts to the consequential fact that *"buying and selling'* — whether of tangible or intangible goods (mechandise, resp. information) — may give, or gives, the impression of being *"instant"*.

[7] adds:

- **Buyer/Seller Relations:** *New forms of connections, interaction between buyers and sellers, and maintaining buyer/seller relationships.*

  Response: Again this simply amounts to extensions to the previously projected and instantiated domain model, ie. to the "emerging requirements". Examples of such requirements could be: (i) Maintainance, on behalf of buyers (typically consumers), and by sellers (typically retailers), records of past and present transactions; (ii) creation of buyer or seller profiles, profiles that summarise strengths and weaknesses of their behaviour; etc.

  Again "the somputable sky is the limit": There are only computability limits to the kind of *"connections, interactions"* and *"relationships"* that can be specified and implemented.

7.2.3 Business/Customer Relations ([7] Sect. 4.0)

[7], as do others, make the following distinctions:

- **B2B • Business to Business**

  Response: Query, order, quotation, confirmation, delivery, acceptance, invoicing and payment relations between enterprises, ie. organisations. Sellers and buyers are businesses. Example: The "pairs" {producer,trader} and {trader,retailer} are B2Bs.

- **B2C • Business to Customer**

  Response: Similar relations, but between enterprises, on one side, and individuals, on the other side. Seller is a business, buyer is a consumer. Example: The "pair" {retailer,consumer}.

- **C2B • Customer to Business**

  Response: Similar relations where the consumer (the buyer) negotiates with sellers, say in a competitive, bidding situation, for price quotations.

- **C2C • Customer to Customer**

  Response: Similar relations, but now between 'consumers' as in Classified Advertisements or Sunday morning flee markets.

7.2.4 E–Commerce more than just Web Design ([7] Sect. 5.0)

- **Web Page vs. "The Heart":**

  E–Commerce is not just designing a web page.

  Response: Note that we have so far said "absolutely" nothing about the man machine interface, hence so far nothing about the web page !

  Instead we have, in line with the DS–9 Call for Papers, emphasised the flow, data and function semantics.

- **A Total Enterprise Re–direction:**

  All parts of the organisation and management of an enterprise (producer, wholesaler, retailer) must be involved in E–Commerce:

  - **Production Department:** *Integration of existing ("traditional") production process with E–Commerce implies a move from push to combined push/pull, from mass production (where products are pushed onto the market by sellers) to flexible and possibly customised production (where buyers request personalised goods).*

    Response: We refer to [11]. In that document we show a model of production planning and execution. The model readily interfaces with E-Commerce as here outlined.

  - **Sales and Marketing Department:** *There will be a shift from traditional mass marketing to one additionally including (E–Commerce's) target marketing.*

    Response: Each such "shift" functionality can be readily incorporated and described according to the principles and technques of the current paper.

  - **Finance and Accounting Departments:** *Payment becomes electronic, and ordering system will be linked to accounting.*

    Response: We refer to [12]. In that document we show a model of banking. The model readily interfaces with E-Commerce as here outlined.

  - **Human Resource Department:** *Retraining of staff due to the above business process re-engineering.*

    Response: The existence of a domain model and of completed requirements specifications suggest that these can be visualised and animated, that is: made into "demos".

    Such domain and requirements demos can be used in educating and in training all stakeholders, not just the retailer, wholesaler and/or producer staff, but also consumers.

    An x–"demo" is some prototype-like software that can visualise, animate, and, in general, abstractly interpret concepts of its universe of discourse x. By visualisation we understand static, two-dimensional images of concepts: Non–temporal diagrams, photos, drawing, pictures, etc. By animation we understand temporal renditions of visualisations: "Cartoon movies", video-clips, etc., even sound. By abstract interpretation we understand any of the above, ie. information presentations, as well as other, more decision or computation oriented functions over concept compositions. Examples could be: *Buyer/seller "what if" scenario generation, buyer/seller interaction flow "bottle–neck" analyses, etc.*

    The borderline between a requirements–"demo" and the actually developed software that meets its requirements is basically a (performance) matter of time and space scale, as well as a matter of pedagogics: Introductory "help" functions, etc.

- **On "Translating" Traditional Business into E–Business:**

  Traditional hierarchical and matrix structured organisations where management served the rôles of (i) regularly, at for example yearly intervals, defining business external aims and customer relations and internal operating procedures, and (ii) resolving ("backstopping") external and internal relation conflicts, need be extended (modified) to continuously monitor & control the external and internal operating procedures.

Response: We refer to [13, 14]. In the first paper and the second report we show how model management and organisation of enterprises. The models given there readily extends to E–Commerce enterprises.

7.2.5 Process–oriented E–Commerce Development ([7] Sect. 6.0)

— A Domain Requirements-oriented Check List

[7] provides a check-list which must influence our view of the semantics of E–Commerce Systems:

(a) *Is the E–Commerce system of type B2B, B2C, C2B or C2C — or which combination thereof ?*

Response: We have earlier indicated that any of these modalities are readily taken care of in the models that can be developed according the the principles and techniques of the current paper.

(b) *Which goods and services are being traded ?*

Response: This is more a question to be posed to the enterprise: Any answer can be readily modelled.

(c) *Which are the market mechanisms ? Will the E–Commerce system*

- *carry out direct selling ?*

  Response: If so, then appropriate E-transactions have to be defined, and 'The Market' must "know" the potential buyers.

- *act as intermediary or as an agent (trader, wholesaler) passing on orders ?*

  Response: If so, then appropriate E-transactions have to be defined, and 'The Market' must be allowed, by sellers, to act as the intermediary.

- *provide full virtual market for sellers and buyers ?*

  Response: See above.

- *act as a factory outlet ?*

  Response: See earlier.

- *provide a fixed pricing structure or permit a request for quotation structure ?*

  Response: If so, then the "Try_to_Match" function mentioned earlier must be appropriately defined.

- *provide an auction or reverse auction facility ?*
  and: define an E–payment mechanism ?

  Response: Etcetera, etcetera: All of these functionalities can and, when present, must be formally defined — at least if one is to follow our dogmas !

(d) *(Related to item (b) above:) Should buy/sell transactions be followed up by cross-selling efforts ?*

Response: See answer to an earlier, similar, question: "If so, then appropriate E-transactions have to be defined, and 'The Market' must "know" the potential buyers."

(e) *Mechanisms for order fulfillment and distribution.*

Response: It was mentioned earlier that one can superimpose a distribution network. Such a network can be likewise formally defined. We refer to two recent student reports, [15, 16] for details of models of logistics. They can be readily interfaced to an 'E-Commerce Market'.

7.3 Discussion

We have, in Section 7.1, touched upon some, but far from all, issues of E–Commerce Semantics. Instead we have been guided, in Section 7.2, by E–Commerce issues as rather systematically enumerated in [7].

Much work remains to be done.

We expect this work to take off in connection with two MSc Thesis projects during the Winter and Spring of 2001.

8 Conclusion

8.1 Discussion

The paper has taken a rather different approach to its target subject: *The software engineering development of E-Commerce systems:*

- Instead of "chasing" what is going on, right now, "out there" in *E–Commerce "land"*, we have stopped up, for a moment of reflections, perhaps a longer moment that most would care for, to try understand the basis on which it might all have started:
  - *The current market*, with its buyers and sellers,
  - to study their ad hoc "protocols", their sometimes professional behaviour, sometimes "sloppy" or "delinquent", yes, at times even "criminal" behaviour,
  - to study their actions, and
  - and to study the information that is part of their interaction protocols and of their internal, albeit informal, perhaps intangible 'operationing'.
- We insist:
  - on not delving into *requirements*, let alone *software design*, before we have at least attained some understanding of the *domain*.

    We may now have, albeit just a rough understanding of the *domain*. But much remains. We have to study, informally describe and formally specify, many more domain attributes, aspects and facets. We elsewhere outline some of the facets: Intrinsics, supporting technologies, management & organisation, rules & regulations, and human behaviour [13]. In this paper we have emphasized intrinsics and human behaviour. And then only to some small degrees.
  - on not delving into *software design* before proper domain, interface and *machine requirements* have been developed [17].

    Domain requirements relate, strongly, to *domain models* through such (formalisable) development ("refinement") relations as are outlined in [17]: projection, instantiation, extension and initialisation.

    We claim that such a *domain requirements development*, with its attendant projection, instantiation, extension and initialisation relations will teach us more about essences of E-Commerce Semantics.itemize
  - We suggest that a suitably "rich", let us call it a generic domain model — such as we have started to develop in this paper — can be *re-used:* Can serve as a basis for the rewriting into slightly or somewhat changed instantiated domain models. An instantiated domain model is one which "mirors", more accurately, a specific customers' problem domain. One for which very specific requirements have to be developed, and one for which a software design has to be assembled, including developed.

- Thus we see two separate "developments": One in which one or more generic domain models are researched and exploratively and experimentally developed, and one in which one or more customer specific, ie. instantiated, domain models are (software technology) developed.

We see our paper contributing to both: (1) To suggest a form of generic domain model, one which is sufficiently loose and thereby general, to serve as a common denominator for several instantiated, domain models; and (2) to outline, together with our other recent papers and lecture notes: [18, 19, 20, 21, 22, 23, 24, 25, 14, 26], how to professionally develop customer specific, scalable software technology.

8.2 Summary

We have presented an approach to E—Commerce software engineeering which:

- `[1] evolves` towards a *Data Semantics* from the basis of an *Information Semantics* that, again, emerges froma *Behaviour* and *Action Semantics;*

- `[2] applies` a "proven technology", that of *Formal Semantics,* to an emerging field — very much in need of precision;

- suggests that one `[3] separates the concerns` of understanding (α) the application *domain* [from before computing & communication is introduced] from understanding (β) the *requirements* [and show how they can be "derived" from a proper domain understanding], and `separates` these again from understanding (γ) the *software design;*

- `[4.1] demonstrates` how to subject the Electronic Commerce application domain to rigorous, formal analysis — and which `[4.2] proposes` the framework for an *E–Commerce Domain Model.*

We submit the above as outlining one proper approach to *the software engineering development of E-Commerce systems.*

8.3 Present and Future Work

Our work has just begun.

We readily admit that we have yet to seriously study the professional, the technical and scientific literature on E–Commerce — other than, of course, the often cited [7]. We are now engaged in a number of E–Commerce R&D projects, in Argentine and in Denmark.

We intend to now spend more time studying, with [7] giving us some steering markers, more aspects of requirements and of possible software architectures and program organisations.

The 'Discussion', Section 8.1, details more of what is to be studied.

8.4 Acknowledgement

We thank T.S.Dillon and his colleagues for having provided [7]. Without its clear composition and its itemisation of E–Commerce issues, we might not have ventured into this emerging software technology area already at this time.

The first author thanks his students, in Denmark, including the co-authors, and his 'students' in Argentine, E. Benegas, M.G. Peralta, C.H. Salgado, and G. Vilallonga, at the National University of San Luis, for their patience in being "served", and commenting astutely on, this paper as it evolved.

9 Bibliographical Notes

References

[1] Norhayati Abd. Mukti and Fong Wing Han. Business to Consumer Trade in Malaysia: Internet Electronic Commerce Model. In *Information Technology and Communication at the Dawn of the New Millenium*, pages 61–74, Light green volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[2] Mark Gabbot. E–Business in South East Asia. In *Information Technology and Communication at the Dawn of the New Milleniu,*, pages 43–52, Pastel/withered green volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[3] Robert Kheng Leng Gray. Application Service Providers (Latest Wave in E–commerce). In *Information Technology and Communication at the Dawn of the New Millenium*, pages 53–62, Pastel/withered green volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[4] Thalerngsak Kijthaweesinpoon and Chris Avram. Technical Options for E–Commerce in Thailand. In *Information Technology and Communication at the Dawn of the New Millenium*, pages 155–164, Aubergine volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[5] Hinny P.H. Kong, L.H. Chen, K.S. Poh, and B.H. Ong. Internet Agents for E–commerce in Travel Industry. In *Information Technology and Communication at the Dawn of the New Millenium*, pages 165–180, Aubergine volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[6] M.I.M. Wahab and Sun Ru. The Impact of E–commerce on Supply Chain Management. In *Information Technology and Communication at the Dawn of the New Millenium*, pages 181–188, Aubergine volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[7] T.S. Dillon, Ronnie Cheung, and Henry Chan. Software Engineering of E–commerce Systems. In *Information Technology and Communication at the Dawn of the New Millenium*, pages 5–15, Light green volume, Bangkok, Thailand, August 1–4 2000. Asian Institute of Technology.

[8] The RAISE Language Group. *The RAISE Specification Language*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.

[9] The RAISE Method Group. *The RAISE Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.

[10] R. Kalakota and A.B. Winston, editors. *Readings in Electronic Commerce*. Addison–Wesley, Reading, Mass., USA, 1997.

[11] Dines Bjørner. Project Information, Monitoring and Control Systems — A Domain Analysis. Technical report, Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, Bldg. 322, DK–2800 Lyngby, Denmark, 1999.

[12] Dines Bjørner, Vasco Rosario, and M. Helder. A Normative Model of Concrete Banking Operations — Banking Rules & Regulations and Staff/Client Behaviours. Research, Department of Information Technology, Software Systems Section, Technical University of Denmark, DK–2800 Lyngby, Denmark, June 1998. (Need be revised: Some typos etc. !).

[13] Dines Bjørner. Domain Engineering, A Software Engineering Discipline in Need of Research. In *SOFSEM'2000*, Lecture Notes in Computer Science. Springer Verlag, 18–24 November 2000.

[14] Dines Bjørner. Domain Engineering, Elements of a Software Engineering Methodology — Towards Principles, Techniques and Tools — A Study in Methodology. Research report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 343, DK–2800 Lyngby, Denmark, 2000. One in a series of summarising research reports [17, ?].

[15] Ths. Hedemand Nielsen and Esben Woods. Container Logistics. Term project report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 322, DK–2800 Lyngby, Denmark, August 2000. A domain model is presented for logistics aspects of contain transport: From source of (customer) origin, by truck, train, air and/or boat, via container yards, etc.

[16] Hans Madsen Petersen. A Logistics System. Term project report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg.322, DK–2800 Lyngby, Denmark, December 2000.

[17] Dines Bjørner. Requirements Engineering, Elements of a Software Engineering Methodology — Towards Principles, Techniques and Tools — A Study in Methodology. Research report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 343, DK–2800 Lyngby, Denmark, 2000. Not yet released. Meanwhile refer to [23]. One in a series of summarising research reports [14, ?].

[18] Dines Bjørner, Souleimane Koussobe, Roger Noussi, and Georgui Satchok. Michael Jackson's Problem Frames: . In Li ShaoQi and Michael Hinchley, editors, *ICFEM'97: Intl. Conf. on Formal Engineering Methods*, Los Alamitos, CA, USA, 12–14 November 1997. IEEE Computer Society Press.

[19] Dines Bjørner. Domains as Prerequisites for Requirements and Software *&c.* In M. Broy and B. Rumpe, editors, *RTSE'97: Requirements Targeted Software and Systems Engineering*, volume 1526 of *Lecture Notes in Computer Science*, pages 1–41. Springer-Verlag, Berlin Heidelberg, 1998.

[20] Dines Bjørner and Jorge Cuellar. Software Engineering Education: The Rôle of Formal Specifications and Design Calculi. Research report, Dept. of Information Technology, Technical University of Denmark, Bldg.345/167–169, DK–2800 Lyngby, Denmark, October 1997. Invited paper for Annals of Software Engineering journal, 1999.

[21] Dines Bjørner. Where do Software Architectures come from ? Systematic Development from Domains and Requirements. A Re–assessment of Software Engneering ? *South African Journal of Computer Science*, 1999. Editor: Chris Brink.

[22] Dines Bjørner. Pinnacles of Software Engineering: 25 Years of Formal Methods. *Annals of Software Engineering*, 2000. Eds. Dilip Patel and Wang Yi.

[23] Dines Bjørner. *Software Engineeering: A New Approach. From domains via requirements to software. Formal specification and design calculi.* Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 322, DK–2800 Lyngby, Denmark, 2000. Presently this document is a rather extensive (approx. 900 page) set of lecture notes. Postscript versions of the document are accessible over the web: `http://www.it.dtu.dk/~db/s2000/notes.ps`.

[24] Dines Bjørner. Domain Modelling: Resource Management Strategics, Tactics & Operations, Decision Support and Algorithmic Software. In J.C.P. Woodcock, editor, *Festschrift to Tony Hoare*. Oxford University and Microsoft, September 13–14 1999.

[25] Dines Bjørner. A Triptych Software Development Paradigm: Domain, Requirements and Software. Towards a Model Development of A Decision Support System for Sustainable Development. In ErnstRüdiger Olderog, editor, *Festschrift to Hans Langmaack*. University of Kiel, Germany, October 1999.

[26] Annabelle McIver and Carrol Morgan, editors. *Programming Methodology: Recent Work by Members of IFIP Working Group 2.3*, chapter Dines Bjørner: "What is a Method ?" — A Study of Some Aspects of Software Engineering. IFIP WG2.3. MacMillan, Oxford, UK, 2001. To be published.

