25

Integrating Enterprise Models and Models for Marketing Analysis

T. Janowski, R. V. Atienza and G. Giménez Lugo
The United Nations University
International Institute for Software Technology
P.O. Box 3058, Macau. E-mail: tj@iist.unu.edu

Abstract

We present an experiment in modelling and analysis of an application domain: competitive manufacturing. The result is a unique formal model which combines previously separate models for marketing (competition) and enterprises (coordination). In particular, we capture formally the marketing mix: product, price, place and promotion, and its effect on the sale of the enterprise. The model is built in stages: market without marketing, marketing without limits and marketing under limited resources. Analysis includes justifying abstraction, down to two enterprises competing for a single consumer.

Keywords

Enterprise Modelling and Integration, Marketing Analysis, Extended Enterprise, Formal Modelling, Property-Preserving Abstraction.

1 INTRODUCTION

The work on modelling of manufacturing enterprises has been traditionally divided between modelling for competition and for cooperation. The first has been exploited for many years now, receiving, not unsurprisingly, a lot of commercial interest and producing many quantitative theories for consumer behavior, market segmentation, demand assessment and forecasting, pricing, advertising and many others (Kotler 1991). Technically, the models are mostly sets of equations (e.g. the equation between price and demand), analyzed using methods from statistic or linear or non-linear algebra (Kotler *et al.* 1988). The second area is more recent,

J. J. Mills et al. (eds.), *Information Infrastructure Systems for Manufacturing II*

appearing soon after CIM itself (Teicholz *et al.* 1987), and aiming at better use of resources and coordination of activities within and between enterprises (Petrie 1992, Vernadat 1996). Here models are more about behaviors than numbers. They describe the resources owned by the enterprise as well as processes which utilize such resources. They own more to methods of specifying and verifying software, more expressive and usually capable to include methods from statistic or algebra. Examples of frameworks for enterprise modelling and integration (often called 'reference architectures') include: GERAM (Bernus *et al.* 1995), CIMOSA (AMICE 1993), GIM (Roboam *et al.* 1989), PERA (Williams 1994), ARIS (Scheer 1992).

This work is a case study in the integration of enterprise and marketing models. There are many reason one may like to seek such an integration: (1) Realistic representation of constraints upon marketing decisions made by the firm: its resources and how they are allocated to business activities. Such constraints are typically written as parameters to marketing equations, unable to capture their true behavioral and dynamic nature. (2) Providing a clear goal to the integration effort, not for its own sake but to compete better with other enterprises. On their own, enterprise models come with no method of assessment of the integration efforts and therefore without a clear sense of direction. (3) Methods to model enterprises are now being extended to collections of enterprises which to some extent cooperate and to some extent compete with each other, forming an extended enterprise. Modelling requires addressing both issues together. We can better exploit opportunities for information-sharing and joint decision-making within the extended enterprise if we can make enterprise models and models for marketing analysis "work together".

There are many open issues when it comes to the integration of enterprise and marketing models. By presenting this case study we hope to put forward and clarify some of them. We focus on one scene: a set of enterprises, all manufacturing one product, competing to take the best share of the demand generated by a set of consumers. Enterprises advertise the attributes of their brands of the product, price among them. Consumers assign weights to the attributes and buy according to the calculated utility of each brand. Within each enterprise decision-making involves the choice of the marketing mix: product, price, place and promotion, under the limitations imposed by the enterprise resources. We consider three stages in the design of the model: (1) Market without marketing: enterprises are able to sell products on the market but cannot by themselves increase the sale. (2) Marketing without limits: enterprises can make decisions by which they could increase the sale, constrained only by their competitors. (3) Marketing under limited resources: enterprises can only make marketing decisions within their capacity to implement them.

The stages are considered in Sections 2, 3 and 4 respectively. Section 5 is about abstraction and Section 6 contains conclusions. Throughout the paper we apply the formal notation of RAISE (Rigorous Approach to Industrial Software Engineering) and its specification language RSL (The RAISE Method Group 1992).

2 MARKET WITHOUT MARKETING

Consider a single product and a number of players on the market for this product, some willing to buy and others to sell. We declare them as abstract types, with functions to buy and sell products (we ignore for now constraints).

```
type
  Consumer, Supplier
value
  buy: Nat × Consumer → Consumer,
  sell: Nat × Supplier → Supplier
```

A market is a set of players where consumers and suppliers can simultaneously sell and buy (one-to-one) products. $\xrightarrow{\sim}$ represents partial functions, **pre** represents a pre-condition, **post** a post-condition, and | is a union of types:

```
type
  Player = Consumer | Supplier,
  Market = Player-set
value
  one_to_one: Nat × Consumer × Supplier × Market ~→ Market
  one_to_one (n,c,s,m) ≡ m\{c,s}∪ {buy(n,c),sell(n,s)} pre {c,s} ⊆ m
```

Consider a transaction between a consumer and a set of suppliers. `Partition` represents different ways to partition demand between suppliers: all partial functions that given a natural number (a demand), a consumer and a market return a map from suppliers to natural numbers (a share of the demand). `Partition` is defined as a sub-type, a type constrained by a predicate `is_wf`. **dom** represents a domain and **rng** a range of a map.

```
type
  Partition' = Nat × Consumer × Market ~→ (Supplier →m Nat),
  Partition  = {| p:Partition' • iswf(p) |}
value
  iswf: Partition' → Bool
  iswf(p) ≡ (∀n:Nat, c:Consumer, m:Market •
            dom p(n,c,m)=suppliers(m) /\ sum(rng p(n,c,m))=n ),
  suppliers: Market → Supplier-set
  suppliers(m) ≡{s | s:Supplier • s ∈ m}
```

A many-to-one transaction is a function with four arguments: demand, consumer, market and partition. It returns a new market by transferring products between suppliers and a consumer, subject to specified partitioning.

```
value
  many_to_one: Nat × Consumer × Partition × Market → Market
  many_to_one(n,c,p,m) ≡ (m\{c}\suppliers(m)) ∪ {buy(n,c)} ∪
                          {sell(p(n,c,m)(s), s) | s:Supplier • s ∈ m} pre c ∈ m
```

Two distinctive ways to partition demand are: equally or in random. Both can be represented by values of type `Partition`, giving rise to the corresponding transactions. But none allows players to influence their share of the demand.

```
value
  equal: Partition • (∀n:Nat,c:Consumer,m:Market •
          (∀ s1,s2:Supplier • {s1,s2}⊆ m ⇒
            equal(n,c,m)(s1) = equal(n,c,m)(s2)) ),
  random: Partition • true
```

3 MARKETING WITHOUT LIMITS

A number of variables controlled by the enterprise affect the level of demand for its products. These called marketing-decision variables are typically classified into four P's (Kotler 1991): product, price, place and promotion. This section provides a simple model for advertising that captures such variables.

3.1 Advertising

Suppose each brand of a product is assigned different attributes like price, reliability, service, efficiency, aesthetics etc. Suppliers advertise the levels of such attributes. Consumers assign weights to them according to how they perceive their importance. Weights and levels let them calculate the relative utility of each brand and therefore partition the demand proportionally.

Formally, assume a type of attributes with one distinguished attribute `price`. Function `attribute` produces the levels of attributes for every supplier, represented by real numbers. Function `weight` produces the weights of attributes for every consumer.

type
Attribute
value
price: Attribute,
weight: Consumer $_{\chi}$ Attribute → **Real**,
attribute: Supplier $_{\chi}$ Attribute → **Real**

Together, the functions allow consumers to calculate the utility of products by different suppliers, by summing up the multiplied levels and weights of all attributes. This may involve some normalizing in order to be able to actually compare values of different attributes, taking some attributes proportionally and some inverse-proportionally (price) etc. In essence, however, we have the definition of the utility function `utility` and function `total` which sums up utilities for the set of suppliers, as below.

value
utility: Consumer $_{\chi}$ Supplier → **Real**
utility(c,s) ≡ sum ({attribute(s,a) * weight(c,a) | a:Attribute • **true**}),
total: Consumer $_{\chi}$ Supplier-**set** → **Real**
total(c,ss) ≡ sum({utility(c,s) | s:Supplier • s ∈ ss})

Then `utility(c,s)/total(c,ss)` is the relative satisfaction of the consumer `c` when purchasing products from the supplier `s`, and given the levels of attributes advertised by all suppliers `ss`. `comp` represents the value of type `Partition` where shares received by different suppliers are in the same proportion as relative satisfaction for their brands.

value
comp: Partition • (∀ c:Consumer, s:Supplier, n:**Nat**, m:Market •
{c,s} ⊆ m ⇒ **let** ss = suppliers(m) **in**
comp(n,c,m)(s)/n = **int** (utility(c,s)/total(c,ss)) **end**)

As an illustrative example from the car industry, take four brands of the Japanese cars: Mitsubishi, Honda, Toyota and Nissan, advertising five attributes of their economy cars: price (in Peso as in the Philippines), reliability (number of "good" cars per 1000), service (number of service centers), efficiency (number of kilometers per liter) and aesthetics (luster of colors). The levels of attributes, their weights assigned by a potential car buyer and calculated utility valuations are below. The resulting market shares between four brands are 27.6, 23.8, 24.2 and 24.4 percent respectively.

Table 1 Attributes, weights and market shares.

	Attributes						
Brand	Price	Reliability	Service	Efficiency	Aesthetics	Utilities	Shares
Mitsubishi	490	986	22	12	9	4.127	0.276
Honda	550	989	12	10	10	3.568	0.238
Toyota	520	990	14	10	9	3.622	0.242
Nissan	470	988	12	12	8	3.656	0.244
				Weights			
	0.84	1.00	0.95	0.70	0.75		

3.2 Segmentation

Advertising may like to address specific class of consumers. This is done by introducing segments into the market. We introduce the type of segments where every player is contained in one segment. The attribute function now takes three arguments: supplier, segment and attribute. The levels of attributes may actually differ for the same supplier between segments.

type
Segment
value
seg : Player → Segment,
attribute: Supplier $_X$ Segment $_X$ Attribute → **Real**

With segments we need a different way to calculate satisfaction of consumers, taking into account the levels of attributes advertised locally. This also affects the definition of the utility function. The new calculation is represented by the value `seg_comp` of type `Partition`:

value
seg_comp : Partition • (∀ c:Consumer, s:Supplier, n:**Nat**, m:Market •
{c,s} ⊆ m ⇒ **let** ss = suppliers(m) **in**
seg_comp(n,c,m)(s)/n = **int** (utility(c,s)/total(c,ss)) **end**),
utility: Consumer $_X$ Supplier → **Real**
utility(c,s) ≡
sum({attribute(s,seg(c),a) * weight(c,a) | a:Attribute • **true**})

3.3 Marketing

The model allows us to define the marketing mix chosen by every supplier: `price` is a real number and reflects the price of the product; `place` is a set of

segments that a supplier targets in its advertising; `product` is a map from the attributes to reals and represents the levels of attributes advertised by default in all segments; `promotion` is a map from segments to reals, representing the price chosen specially for those segments (smaller than `price`):

type
  Marketing'::
    price: **Real**
    place: Segment-**set**
    product: Attribute $\xrightarrow[m]{}$ **Real**
    promotion: Segment $\xrightarrow[m]{}$ **Real**
  Marketing = {| x:Marketing' • iswf(x)| }

value
  iswf: Marketing' → **Bool**
  iswf(x) ≡ price ∈ **dom** product(x) /\
    price(x) = product(x)(price) /\ **dom** promotion(x) = place(x) /\
    (∀ s:Segment • s ∈ place(x) ⇒ promotion(x)(s) < price(x))

4 MARKETING UNDER LIMITED RESOURCES

So far the value of sale by suppliers was only constrained by their competitors. We now capture constraints that are the result of internal limitations. Two are taken into account: how many products a supplier is able to sell (the stock) and how many products a consumer can afford to buy (the budget).

4.1 Limited Stock

We assume that all players include a stock, a function that given a player returns the number of products it currently owns. Operating on the stock are usual functions for buying and selling.

value
  stock : Player → **Nat**
axiom
  (∀ c:Consumer, n:**Nat** • stock(buy(n,c)) = stock(c)+n),
  (∀ s:Supplier, n:**Nat** • stock(s) ≥ n ⇒ stock(sell(n,s)) = stock(s) − n)

The definition of stock will affect transactions on the market. The stock of the supplier must be rich enough to fulfill the consumer's demand:

```
value
  one_to_one: Nat × Consumer × Supplier × Market → Market
  one_to_one(n,c,s,m) ≡
      m\{c,s} ∪ {buy(n,c), sell(n,c)} pre {c,s}⊆ m /\ stock(s) ≥ n
```

We need similar changes to many-to-one transactions. This takes a demand, a consumer, a partition and a market, and produces a new market. The result is unspecified if the calculated share exceeds the stock of even a single supplier.

```
value
  many_to_one: Nat × Consumer × Partition × Market → Market
  many_to_one(n,c,p,m) ≡

let c' = buy(n,c),
    ss' = {sell(p(n,c,m)(s),s) | s:Supplier . s ∈ m}
in m\{c}\suppliers(m) ∪ {c'} ∪ ss' end
pre c ∈ m /\ (∀ s:Supplier . s ∈ m ⇒ stock(s) ≥ p(n,c,m)(s))
```

4.2 Limited Budget

We also assume that all players on the market include a budget, a function that given a player returns the value of money that the player currently owns. Operating on the budget are the functions for debiting and crediting players.

```
value
  budget: Player → Nat,
  credit: Nat × Supplier → Supplier,
  debit: Nat × Consumer ⇒ Consumer
axiom
  (∀ s:Supplier, n:Nat . budget(credit(n,s)) = budget(s)+n ),
  (∀ c:Consumer, n:Nat . budget(c)≥n ⇒ budget(debit(n,c)) = budget(c) - n)
```

Every transaction will debit the consumer and credit the supplier, the price as advertised in the consumer's local segment. A supplier must have enough products to fulfill the demand, a consumer enough money to pay.

```
value
  one_to_one: Nat × Consumer × Supplier × Market ~→ Market
  one_to_one(n,c,s,m) ≡
    let p = int attribute(s,seg(c), price)
    in m\{c,s} ∪ {buy(n,debit(n*p,c)),sell(n,credit(n*p,s))} end
    pre {c,s}⊆ m /\ stock(s) ≥ n /\
      budget(c) ≥ n * int attribute(s,seg(c), price)
```

For many-to-one transactions, the amount of debit incurred by consumers must be calculated based on the share of the demand allocated to individual suppliers and the price advertised by each of them in the consumer's segment. Function `deb` calculates this value. Likewise, `cred` calculates the value of credits for individual suppliers:

```
value
  deb: Nat × Consumer × Partition × Market ~→ Nat
  deb(n,c,p,m) ≡
    sum({p(n,c,m)(s)* int attribute(s,seg(c),price) | s:Supplier • s ∈ m})
    pre c ∈ m,
```

```
cred : Nat χ Consumer χ Supplier χ Partition χ Market → Nat
cred(n,c,s,p,m) ≡ p(n,c,m)(s) * int attribute(s,seg(c),price) pre {c,s} ⊆ m
```

Suppliers must have enough products to fulfill the allocated share of the demand, a consumer must have enough money to pay for its demand, taken into account the prices of individual suppliers:

```
value
  many_to_one: Nat × Consumer × Partition × Market ~⇒ Market
  many_to_one(n,c,p,m) ≡
    let
      r = p(n,c,m), c'=buy(n,debit(deb(n,c,p,m),c)),
      ss' = {sell(r(s), credit(cred(n,c,s,p,m), s)) | s:Supplier • s ∈ m}
    in
      m\{c}\suppliers(m) ∪ {c'} ∪ ss' end
    pre c ∈ m /\ budget(c) ≥ deb(n,c,p,m) /\
      (∀ s:Supplier • s ∈ m ⇒ stock(s) ≥ p(n,c,m)(s))
```

5 MARKETING ABSTRACTION

Sections 2, 3 and 4 presented a sequence of increasingly elaborate models for enterprises competing on the single product market. We now show how to reduce the complexity of such models, therefore any analysis which is based upon them, applying abstraction. For simplicity we concentrate on the market without segments and its corresponding partition value comp. We introduce abstraction in two stages: (1) reducing a set of consumers into one and (2) reducing a set of suppliers into two, competing with each other. Abstraction preserves the share of the demand by suppliers.

5.1 Consumer Abstraction

Consumer abstraction is a function which produces a single consumer from the set of consumers, by adding up the weights for all attributes:

```
value
  cabs: Consumer-set → Consumer
  cabs(cc) as c' post
   (∀ a:Attribute • weight(c',a) = sum({weight(c,a) | c:Consumer • c ∈ cc}))
```

We would like this function to preserve the shares: the sum of shares received by every supplier from individual consumers equals the share they receive from the abstracted consumer, with respect to the sum of the original demands. However, this property is somehow optimistic given that our abstraction removed all distinction between consumers in terms of weights assigned to the attributes. As it turns out, this property is indeed true if we take demands by individual consumers which are inverse-proportional to their total utility values (for given set of suppliers on the market), say obtained by multiplying the total utility value by some common constant. This is written down as the following axiom which we can prove from the definition above:

```
  axiom
   (∀ cc:Consumer-set, d:Nat •
    (∀ m:Market, s:Supplier • s ∈ m ⇒
     let ss = suppliers(m) in
      comp(d*int total(cabs(cc),ss),cabs(cc),m)(s) =
      sum({comp(d*int total(c,ss),c,m)(s) | c:Consumer • c ∈ cc})
     end ) )
```

Then it remains to show that the total utility value for the abstracted consumer equals the sum of such total utilities by individual consumers.

```
axiom
 (∀ cc:Consumer-set, m:Market •
  let ss = suppliers(m) in
   total(cabs(cc),ss) = sum({total(c,ss) | c:Consumer • c ∈ cc})
  end )
```

This justifies preservation of shares by consumer abstraction, provided consumer demands are inverse proportional to their total utility values.

5.2 Supplier Abstraction

This step is to reduce the number of suppliers to two. One supplier is chosen to identify "our" enterprise. The rest is competition which we want to reduce into one supplier. This reduction should not change the share of consumer demand: the sum of what individual suppliers were able to sell before reduction should be the same as what the abstracted supplier is able to sell afterwards.

```
value
  sabs: Supplier-set × Supplier ⇒ Supplier
  sabs(ss,s) as s' post
    (∀ c:Consumer, n:Nat • comp(n,c,{s,s'})(s')=
      sum({comp(n,c,ss ∪ {s})(t) | t:Supplier • t ∈ ss})
      ) pre s ∉ss
```

We can achieve this effect by adding up the levels of attributes. The result is similar like before: for any consumer, the utility value of the new supplier equals the sum of utilities of all suppliers in the set. Unlike before, the shares before and after abstraction are "preserved" for any level of demand.

```
axiom
 (∀ ss:Supplier-set, s:Supplier • s ∉ ss ⇒
  (∀ a:Attribute •
   attribute(sabs(ss,s),a) = sum({attribute(t,a) | t:Supplier • t ∈ ss})))
```

6 CONCLUSIONS

We presented a sequence of models of enterprises competing in a single-product market: (1) market without marketing - enterprises cannot by themselves increase the sale of their product; (2) marketing without limits - enterprises can increase the sale by advertising product attributes; (3) marketing under limited resources - enterprises can only make decisions within their capacity to implement them. Presented models are more on the side of competition than coordination: modelling for marketing. However, only lack of space prevented us from

considering more involved descriptions of the enterprise, similar like in (Janowski *et al.* 1998) and extending Section 4. The paper achieved the basic frame for describing formally enterprises which exist on the market, their marketing decisions and the effect of such decisions on the sales.

There are a number of directions we plan to proceed in future. One is to make the models more concrete, say to consider a multi-product market and ways to express and carry out production: assembly of a product from a number of subproducts. Another direction: although the models presented in this paper seem to illustrate that the marking decisions are made subject to existing resource constraints, it remains possible that the models allow the enterprise to actually determine the resource requirements of a chosen marketing mix. Last but not least, we would like to study the models in this paper as a way to give formal semantics to an application-specific language for modelling and formal analysis of competing (on the market) as well as cooperating enterprises.

REFERENCES

AMICE (1993) *CIMOSA: Open System Architecture for CIM.* Springer Verlag.

Bernus P. and Nemes, L. (1995). Enterprise Integration - engineering tools for designing enterprises, in *Modelling and methodologies for Enterprise Integration* (eds. P. Bernus and L. Nemes), IFIP WG 5.12, Queensland.

Janowski, T., Giménez Lugo, G. and Zheng, H. (1998) Composing Enterprise Models: The Extended and the Virtual Enterprise, in *Intelligent Systems for Manufacturing* (eds. L. M. Camarinha-Matos, H. Afsarmanesh and V. Marik), IFIP, Chapman and Hall.

Kotler, P. and Lilien, G. (1988) *Marketing Decision Making: A Model-Building Approach*, Harper and Row Publishers.

Kotler, P. (1991) *Marketing Decision Making*, Harper and Row Publishers.

Petrie, C. (1992) *Enterprise Integration Modeling*, (ed. C. Petrie), MIT Press.

Roboam, M., Zanettin, M. and Pun, L. (1989) GRAI: Integrated Methodology to Analyse and Design Manufacturing Systems. *Computer-Integrated Manufacturing Systems*, **2**, 82-98.

Scheer, W. (1992) *Architecture for Integrated Information Systems.* Springer Verlag.

Teicholz, E. and Orr, J. (1987) *Computer Integrated Manufacturing Handbook*, McGraw-Hill Company.

The RAISE Method Group (1992). *The RAISE Specification Language*, Prentice Hall.

Vernadat, F. (1996). *Enterprise Modelling and Integration*, Chapman and Hall.

Williams, T. (1994) The Purdue Enterprise reference Architecture, *Computers in Industry* **24 - 2**, 141-158.

7 BIOGRAPHY

Tomasz Janowski is a Research Fellow of UNU/IIST. He received an MSc in Mathematics from the University of Gdansk (Poland) and a Ph.D in Computer Science from the University of Warwick (England). His research interests include logics for provable fault-tolerance, real-time scheduling, formal models for manufacturing and the integration of formal and informal techniques in software development.

Rumel V. Atienza is with De La Salle University, Manila, The Philippines. He received BSc in Mathematics from the University of the Philippines and MBA from De La Salle University where he is currently a candidate for the Doctor of Management degree, with interest and professional experience in marketing management. He was a fellow at UNU/IIST between September and December 1996.

Gustavo Giménez Lugo is a Fellow of UNU/IIST. He received his MSc in Industrial Informatics from the Federal Centre for Technological Education of Paraná State (CEFET-Pr/CPGEI), Curitiba, Brazil where he is a research assistant. His interests are focused on formal modelling and analysis of production systems.