New Web Application Technologies for Global Decision Support in Sales and Marketing - Guidelines for Model Builders

Mihai Calciu
Dan Somnea

We introduce a new stylised simulation model that combines transactional and relationship marketing aspects. A multi-branded offer from multiple companies that generates attraction and retention potential is confronted to a multi-segmented demand with specific dual response behaviour. Dynamic market transitions model loyal and versatile customer flows in time. The model is implemented as an on-line marketing simulation and decision support system. Based upon this simple model we discuss how a progressive modelling approach that uses object orientation can help decompose and recompose marketing problems by managing complexity in order to produce realistic modelling solutions and accelerate their advance to the market. Some implied model and decision support building frameworks are discussed. Several traditional web solutions are revisited, applied and compared to some more recent XML based semantic web technologies. New ways of embedding models (models as documents, models as services etc.) are explored.

Key words: *marketing modelling, decision support, decision calculus, object orientation,internet*
JEL classification code: M15, M31

Introduction

Despite a long tradition in quantitative research in marketing and important accumulations of models, little diffusion of models and their applications can be observed.

As to Little (1970, p. B-466) "the big problem with models is that managers practically never use them". This situation seems not to have changed a lot ever since. "Even several decades after the earliest operational marketing models were first introduced, their impact on practice remains far below its potential" (Eliashberg & Lilien, 1993, p.19). "Many fewer models are actually used than are developed" (Lilien & Rangaswamy, 2000, p.233). The most frequently mentioned causes for this low adoption of marketing models are low productivity in building and implementing models and bad communication between researchers and mangers.

Building problem specific models and making them operational still remains not very productive, the required effort in order to obtain useful results seems excessive.

Multiple qualifications are needed, as these models have a triple representation (Geoffrion, 1987) a natural one, convenient to the communication with modelling non-specialists, a mathematical one, suited for development and analytical use and an "informatics" or computer executable one.

The lack of interest exhibited by managers, that has been often mentioned, is due to the non understanding of models and to a dependency perception of the manager towards the

analyst. The sentiment of dependence makes the manager feel uncomfortable because less powerful.

Another difficulty is based on the fact that managers often feel that the analysts have insufficient knowledge about their field) .

The manager's lack of comprehension is often due to bad communication with analysts who are perceived as too "techno-centric" instead of being "problem centric".

Although Marketing Decision Support Systems (MDSS) approaches have always been shaped by IT advances as can be seen from the Wierenga and van Bruggen (1997) classification of MDSS, Marketing scientists seem to give little importance to information technologies (IT).

Newer IT based data collection and modelling techniques have taken too much time to be adopted placing marketing scientist in great risk to become marginalised as thought leaders in these fields. Tracking customer behaviour over the Internet or recording such behaviour through loyalty cards and real world experiments observing such behaviour are largely computer science dominated. Newer modelling techniques, such as Bayesian networks, neural networks, and data mining have also been actively developed and tested in other areas before being embraced by marketing modellers.

Most traditional marketing models have neglected factors that enhance how the models will be used. "Increasingly, models that do not design in features that take advantage of the distributed and data-rich context provided by the Internet ... will become irrelevant: they will not get used, and will have dimi-

nished importance to future developments in the modeling field. To develop models that do get used, modelers must pay attention to the IT-infrastructure under which their models will be used." (Lilien & Rangaswami, 2000,p232)

This paper discusses and illustrates the possibilities of newer modelling approaches, and decision support designing and development technologies and tries to familiarise marketing modellers with new IT infrastructures for distributed applications. In this way we try to help "retooling: traditional marketing modelers" in order to get their models used on a larger scale.

We introduce a new stylised simulation model that combines transactional and relationship marketing aspects. A multi-branded offer from multiple companies that generates attraction and retention potential is confronted to a multi-segmented demand with specific dual response behaviour. Dynamic market transitions model loyal and versatile customer flows in time. The model is implemented as an on-line marketing simulation and decision support system. Based upon this simple model we discuss how a progressive modelling approach that uses object orientation can help decompose and recompose marketing problems by managing complexity in order to produce realistic modelling solutions and accelerate their advance to the market. Some implied model and decision support building frameworks are discussed. Several traditional web solutions are revisited, applied and compared to some more recent XML based semantic web technologies. New ways of embedding models (models as documents, models as services etc.) are explored.

A stylised marketing model

While learning about new information technologies like database or spreadsheet software we often get some boring management examples with employees, departments etc. why not replacing such entities with more familiar and actionable ones like brands, segments, markets etc. and learn how IT can facilitate their interaction within the framework of a simple marketing model.

We introduce a stylised minimalist marketing simulation model combining transactional and relationship marketing, the two paradigmatic dimensions coexisting in nowadays marketing strategy.

In this model, transactional marketing mix is represented by brand positioning and repositioning strategies supported by traditional communication activities. The market is a perceptual space governed by gravity laws. Brands are positioned and exert attraction upon customer segments who have mass and "ideal point" positions in the market space.

Relationship marketing mix is represented by interactive marketing activities (e-commerce and/or direct marketing), by customer retention and loyalty programmes aimed to increase product/brand quality and indirectly customer satisfaction or to increase switching costs (loyalty cards, frequent flyer programmes etc.). A comprehensive review of loyalty programmes can be found in Meyer-Waarden and Benavent (2001).

Each positional segment has relationship sub-segments (key and non-key customers) with variable responsiveness to

transactional and relationship mix efforts. Key customers tend to be more sensitive to relationship mix incentives than non-key.

Market share results from a subtle transition mechanism governed by attraction and loyalty: new customers come into the market, others leave the market, some become loyal and others turn "versatile".

Market Segments response to marketing efforts is dual. It results in loyalty and perceived attractiveness.

Traditional, transactional marketing mix activities generate primarily attraction while newer relationship marketing mix is supposed to generate customer retention and loyalty.

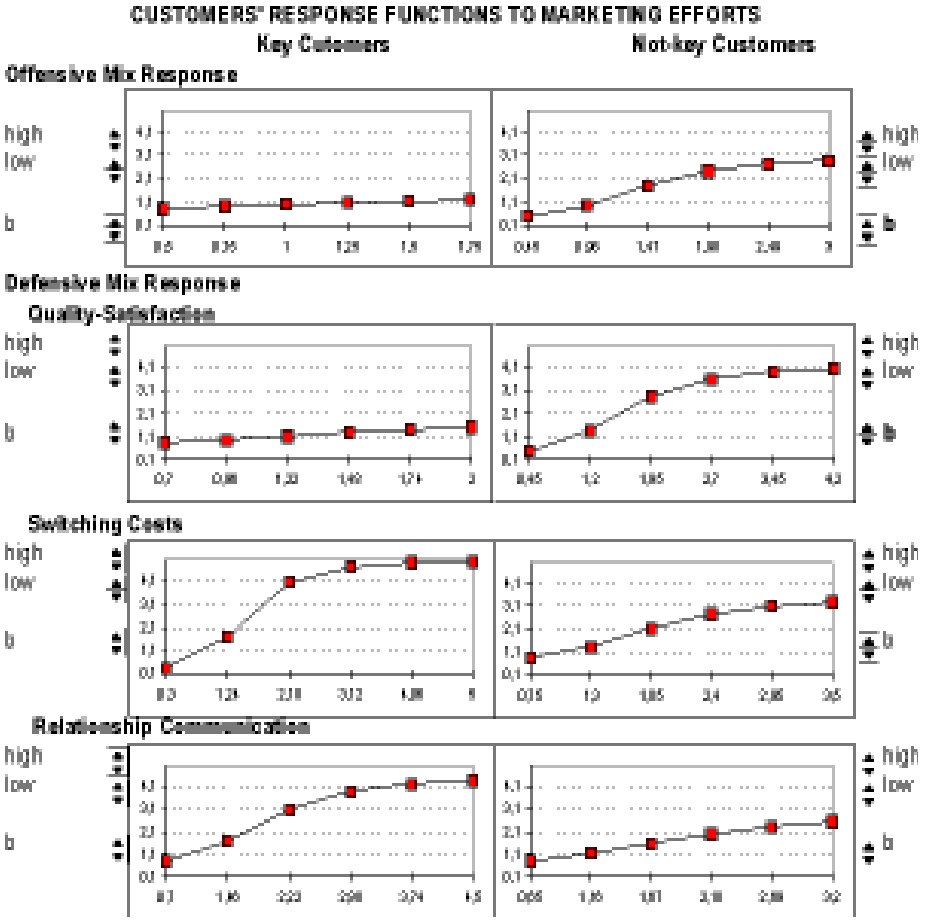
Loyalty toward a brand is measured as the proportion of "hard core loyal" customers. The distinction between "hard core loyals" and "potential switchors" was first used in marketing by Kuehn (1961) and more recently by Colombo and Morrison (1989) and Bultez (1996, 1997)

Key and non-key segments have their own response function and varying reactivity to marketing mix efforts. The reactivity to "retention mix" is modelled to be stronger than the reactivity to offensive marketing. It uses the generally accepted assumption that it is less costly to keep existing customers than to attract new ones.

The market response functions are S-shaped and use relative input. This means that, compared to a previous period, one can estimate the needed effort in order to maintain the same market response and one can identify the lower and higher

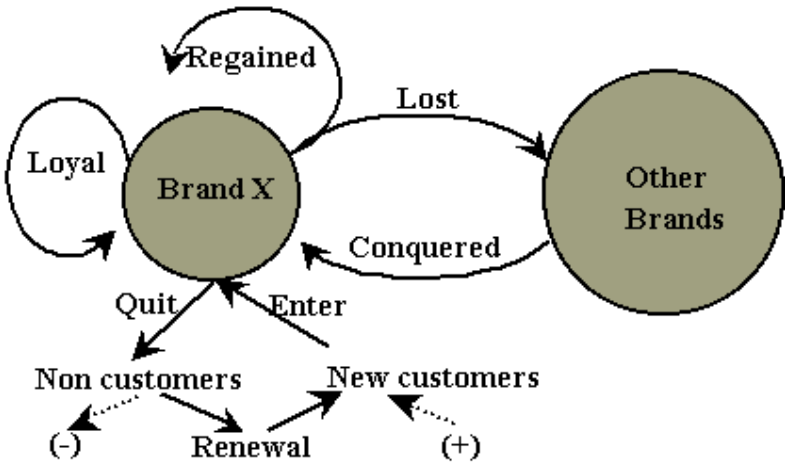
limit within which the market response can vary given different relative amounts of marketing effort. Response parameters and functions vary by segment and mix element according to a given scenario (Figure 1)

Figure 1. A scenario with varying response functions for key and non-key customers



The customer flows from one period to the other between a given brand and the market are shown by the transition diagram in Figure 2.

Figure 2. Customer flows from and to a brand

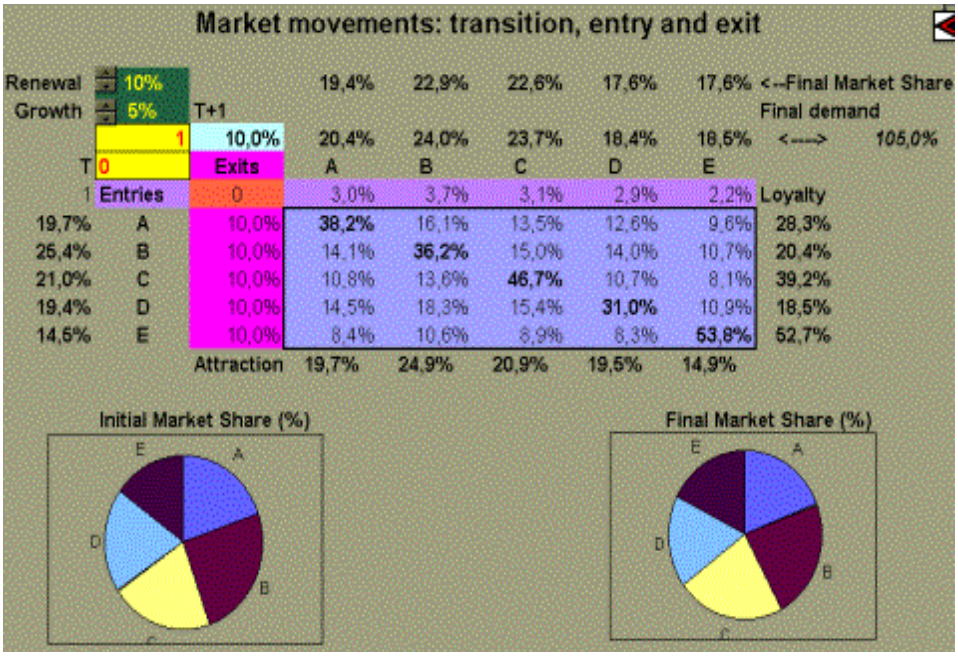


There are two kinds of repeat buyers: hard core loyal customers and regained switchors. The customers gained from other brands and those lost to other brand are also considered to be switchors. Market entry and exit is modelled using a combination of market specific renewal and growth rates. The renewal rate is constant and indicates the number of quitting customers that are replaced by new ones as compared to the total number of customers. The growth rate can be positive when additional customers enter the market and negative when additional customers leave the market. Both renewal and growth rate are exogenous values, given in a predetermined scenario.

The transitions mechanism on the market (or market segment) is controlled by a transition matrix like the one in Figure 3.

Transitions are governed by retention and attraction forces generated by brands' transactional and relationship marketing mix efforts.

Figure 3. Customer transitions determined by brands' attraction and retention, and market entry and exit rates.



In order to distinguish market entry and exit from customer transitions between and within brands, the transition matrix is divided into four areas. They are highlighted with different colours. The same division is maintained in the formal representation of the same matrix given in equation (1). Market entries are controlled by the first row, a vector recording the

part of new customers attracted by each brand. The first column controls the proportion of customers that are leaving the market. The remaining matrix defines the transitions between and within brands. The main diagonal of the latter matrix contains customer retention probabilities which consist of loyalty (the part of hard core loyal customers) and the part of regained switchers as suggested by Bultez (1996, 1997).

$$P = \begin{pmatrix} 0 & [b + (1-d)g]a' \\ b - dg & [1-(b-dg)][F + (1-f)a'] \end{pmatrix} \qquad (1)$$

where: b = customers' renewal rate; g = customers' growth rate ; d = dummy variable that is one when g <= 0 and zero otherwise; **F** = diagonal matrix of loyalty rates; **f** = vector of loyalty rates ; **a** = attraction vector.

The proportion of "hard core loyal" customers a firm has in each segment is given by the loyalty index. It depends on the defensive mix it opposes to other brands. The relative attractiveness of the competing firms on each segment (at market level) determines the share of switchers and new customers it obtains. The variations in market share and in customer structure from one period to the other determine overall gains.

Market transitions represented above are operated at segment level. They can be seen as within segment transitions as there is no exchange with other segments. RFM like transitions dear to direct marketing are typically between segment transitions. In this model between segments transitions are limited to relationship sub-segments (key and non-key customers) and are implemented in a simplified way. They are predeter-

mined by the market environment (scenario induced) and cannot be influenced directly by interactive marketing activities. It is relatively easy to change this exogenous transition mechanism to an endogenous one where transitions from key to non-key are described by a customer migration tree and governed by probabilistic response rates, that can be influenced by direct marketing activities. We have here a loose integration point with more specialised marketing fields like direct marketing which shows the flexibility of this modelling approach.

While extremely stylised and generic and primarily a simulation model, if connected to real data from panels a model of this kind can easily become a strategic decision support model. We think that such a model can serve as common denominator for marketing scientists who wish to better understand and use IT in order to accelerate the diffusion and adoption of models. It can help match our domain specific vocabulary with IT terminology and reveal the utility of certain technological aspects.

- ***Progressive modelling philosophy and object orientation***

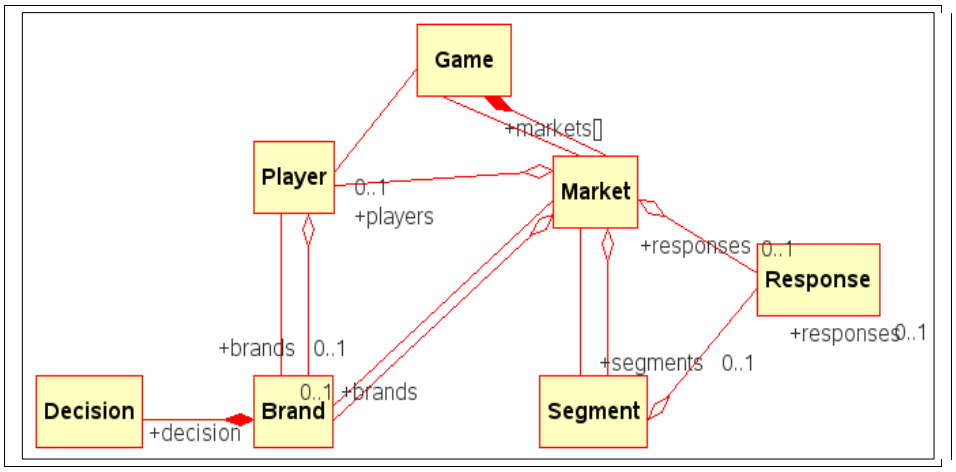
Before discussing technology it is useful to understand some conceptual frameworks that advocate flexibility and facilitate model implementation.

Historically decision calculus (Little, 1970) is the first such framework and probably the only one originating from the field of marketing. Decision calculus is a model building philosophy that fixes a set of conditions that models have to sa-

tisfy in order to be used by managers. It insists mainly on model formulation aspects. Models should allow subjective parameter estimation by managers (see figure 7). Models must be "simple, robust, easy to control, adaptive, as complete as possible and easy to communicate with" (Little, 1970, p. B-466). Robustness applies to a model if it is difficult for a user to obtain bad answers. A model must start simple and easy to control in order to be understood and to attract the managers interests. This does not mean that the model should be simplistic, it still has to be a good abstraction for the real problem and it must be able to evolve and grow in complexity.

Object orientation is a way to analyse and build complex systems and decompose them into logical (classes and objects) and physical (processes module architectures) models with their static and dynamic interactions. It is largely used in computer science and became standard on the World Wide Web. As a systems analysis tool it marked an important evolution compared to the traditional procedure oriented approaches. A system like the one evoked here can be decomposed into a reduced number of categories or classes (economy, market, firm, brand, customer segment etc) using abstraction (highlighting essential properties) , encapsulation (hiding detail), modularity and hierarchy. The class diagram in Figure 4 shows the classes that make up our system and how they relate to each other. It shows which classes “know” about which classes or which classes “are part” of another class.

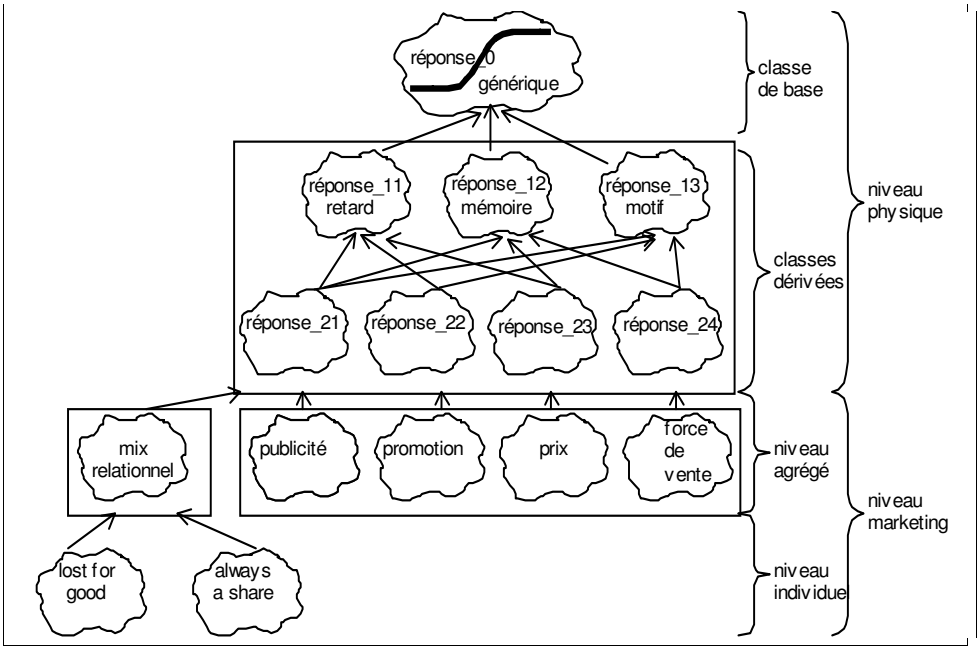
Figure 4 - Class diagram of a stylised marketing system



Each class defines the attributes and the methods of a set of objects of the same kind. Objects are realisations (instantiations) of these classes.

Response is central to marketing, each marketing action aims to generate response at "individual" segment level in terms of attitudes, preferences, buying intentions or at market level in terms of market share. The modelling of response offers the occasion to apply and illustrate a special kind of hierarchy based on inheritance in which more complex classes evolve from simpler ones by inheriting their properties and methods and adding new ones. It mimics an evolutionist complexification process as illustrated in figure 5.

Figure 5 – Inheritance based hierarchy of response models

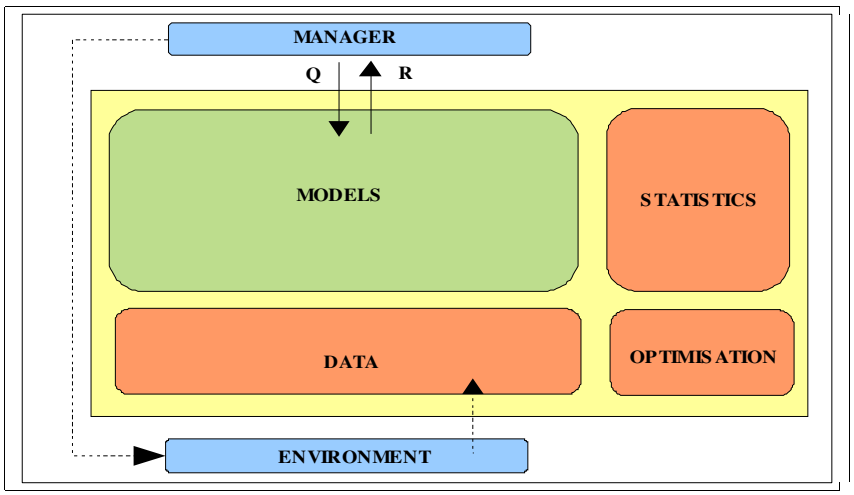


Simple response models evolve towards more complex ones by adding dynamic effects like temporal delay and motifs or memory effects. These specialised response classes combine further in the hierarchy to produce distinct response classes (models) for each marketing mix element. For the relationship mix this evolutionary process continues at individual customer level by adding behavioural dynamics ("lost for good" and "always a share").

- ***Choosing a Marketing Decision support development and implementation framework***

One of the best known marketing DSS frameworks was presented by Little (1979), it is based on the so called DDM (Dialogue, Data, Model) paradigm of Sprague and Carlson (1982) that formed the dominant architecture for DSS from the sixties throughout the eighties (Eom, 1995). Little's framework that we adapt here for web based decision support (see Figure 6) has besides the interface with the manager, four main components: models, data, statistics and optimisation. As oposed to the model base that in our view should mainly include DSS models solving semistructured and unstructured marketing problems, the statistics and optimisation component can be seen as regrouping solutions to structured problems.

Figure 6 – Marketing Decision Support System Framework (Little, 1979)



The **web** is at this moment **one of the best accepted interfaces by users**. Managers like to see DSS implemented in environments they are familiar with. It's not rare to see people wasting hours trying to solve problems on spreadsheets, that could have been instantaneously solved using more specialised statistical software. The web is probably the software application with the largest and fastest adoption ever and with which managers are most familiar. Besides making the Internet usable by non-scientists the Web was noteworthy in its author's view (Tim Berners-Lee) for turning what had been a rigidly hierarchical tool (digital computers) into a device that may eventually make the sort of loose associations that our brains do. "the Web and the HyperText Transfer Protocol - HTTP – that underlies the communication of data on the Web have become a vital part of our information network and day to day environment." (Lang, 2007, p.1). Estimates of the proportion of data sent via HTTP range from 31% to 75% Claffy and Miller (1998), Spr (2004). This is very convenient if managers want to use their own data when they engage in getting decision support over the Internet.

Optimisation and statistics are solving components for structured problems. In the field of structured problems modelling linked to OR-MS real modelling languages like GAMS, AMPL et SML have been developed. They help construct and articulate models into sophisticated compositions based upon convenient, intuitive and mathematically well defined model representation schemes (Huh, 1993). Similar approaches lead to specialised programming languages for fields like mathematics (Matlab) and statistics (S-language). A cen-

tral idea of these evolutions in order to make modelling more flexible is separation between model structure, data structures and solvers. Solvers who concentrate on the procedures and algorithms to be invoked by models in order to solve a problem, are kept in the backend. Such systems and their solvers progressively integrate newer modelling techniques, such as Bayesian networks, neural networks, and data mining that are better suited to deal with the explosion in data generated by the relationship marketing systems.

The data framework component has also marked significant progress. Advances in Data Base technology has resulted in very stable, powerful, secure, client-server and sometimes object oriented database systems. Through database connection drivers (ODBC, JDBC ..) they seamlessly integrate with the model base. Additionally Datawarehouse capabilities with OLAP, MOLAP help aggregate data prepare them for statistical treatments. Customer decision support systems over the Internet and agents use such OLAP engines in order help customers make their choices, compare products and prices over the Internet. All these developments are deemed to deal with the data flood generated by recording mostly behavioural data generated by e-commerce, direct marketing and loyalty card systems.

The decision support modelling component is probably the most complex. It must support and not replace the decision-maker, be user friendly and interactive, regroupe data and methods to solve semistructured problems. Enterprise Information Systems or legacy systems are to rigid to respond to the fluctuating ad-hoc needs of marketing analysis (Hogue

& Greco, 90). "Domain-specific or specialized software such as S, Matlab simply because the overhead of learning the philosophy, syntax and nuances of these languages are excessive for casual use" (Lang, 2001,p.2). They are better suited for structured problems than for semi-structured or unstructured problems facing marketing decision support modelling. Therefore we suggest that decision support implementations should follow mainstream advances in computer applications development.

- ***A Java based framework for MDSS development and implementation as distributed objects web applications***

While there is a certain convergence in the evolution of computer application development and implementation platforms the technologies used by them are rather different and complex. Therefore adopting such a platform is a long term choice. Our choice is guided by the need to favour MDSS diffusion and our criteria are openness, availability, portability and recognition by the computer science profession.

Openness, the possibility to have clear insight on software building blocks, is extremely important for the whole Internet community who doesn't like to see parts of this interconnected world be colonized by proprietary software and locked up in software patents percieved as bariers to rapid software evolution. Openness is also an argument for managers who don't like to be tied up in their relation with the marketing analyst to adopt MDSS. Availability is the possibility to use

the product (and its updates as soon as they are launched). Portability regards the resulting DSS that should function on different hardware or be easily transformed in order to achieve this goal. We think that these criteria are best met Java Web solutions.

An old dream of computer scientists and by derivation of marketing scientists was making programs and decision support solutions portable and compatible to a whole diversity of existing machines. Therefore any innovative solution in this direction gained great respect and was onored by the most prestigious prices[1]. An important step was the introduction of of C, a high level programming language (Kernigan & Richie, 1978) particularly close to the machine language (machine code ..) and therefore producing very fast programs. The Unix computer system that was entirely written in C was the first system that could be ported to any machine in the world due to the fact that C compilers were available on almost all computer systems. It's almost exclusively on Unix machines that the Internet (with its TCP/IP protocol ..) was born and matured until the beginning of the nineties when personal computer users became massively interested by it's last service the world wide web (Berners-Lee, Cailliau, Pellow & Secret, 1993). As these non-Unix systems (Windows, Mac-Os[2]) were sharing programmes in this interconnected world a new kind of compatibility was required. The solution came from Java both a programming language and a platform. Most plat-

1 The Association for Computer Machinery (ACM) honoured with its prestigious award: Unix (1983), TCP/IP the bases of Internet (1991), the World Wide Web (1995) and Java (2002).

2 Apple gave up it's non-Unix sytem in order to adopt Unix.

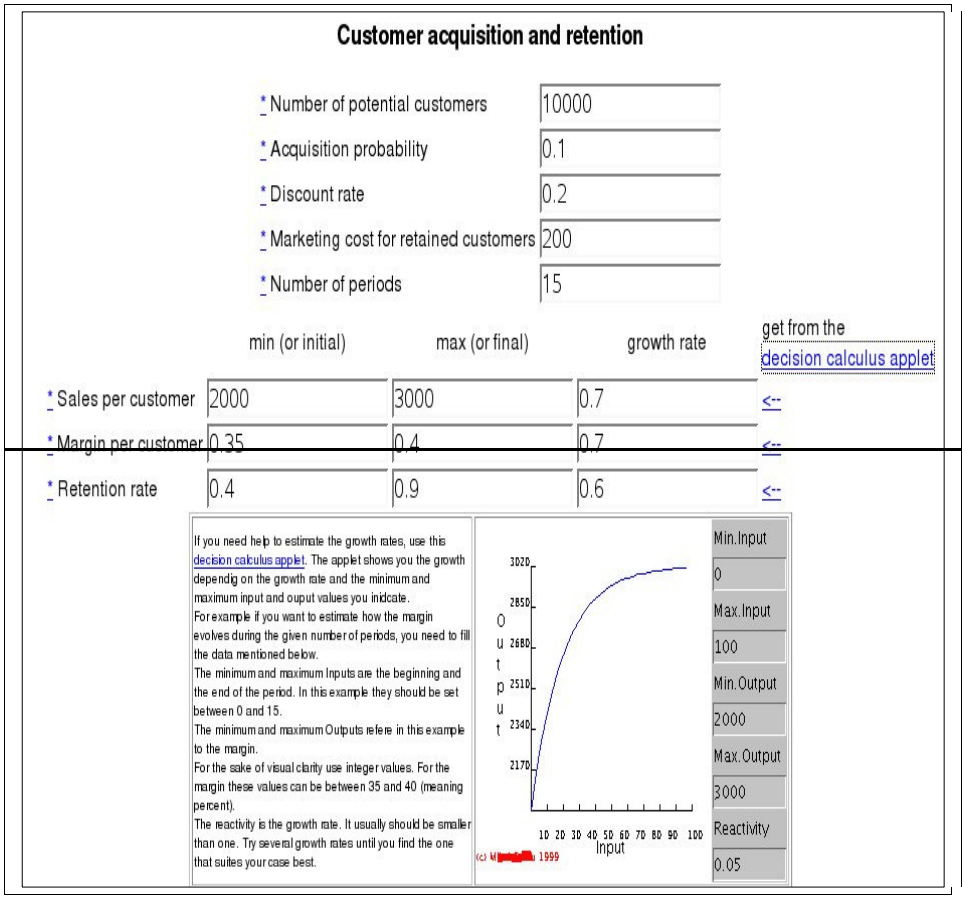
forms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware platforms. It's a kind of virtual computer that can be implemented on almost any computer and also on other devices like household devices (refrigerators, hovers etc.. which was the initial purpose of the Java project) etc. Programmes written in Java get compiled into bytecode, the machine language of the Java Virtual Comupter (or JRE, or Java Interpreter) who then at execution time translates each line of code into the machine code of the specific hardware platform.

Marketing DSS "once written" in Java can be "run anywhere". The Java programming language, with its portability, object-oriented model, support for multithreading and distributed programming, and garbage collection features, is becoming the language of choice for the development of large-scale distributed applications. (Kazi et al., 2000)

Applets, small client applications that used the Java virtual machine installed in the web browser, were probably the first notorious manifestation of Java technology.

We used applets in a web page in order to give managers the opportunity to visually indicate customer response to acquisition and retention efforts by subjective estimation or decision calculus as in figure 7. In exchange they obtained Customer Equity calculations.

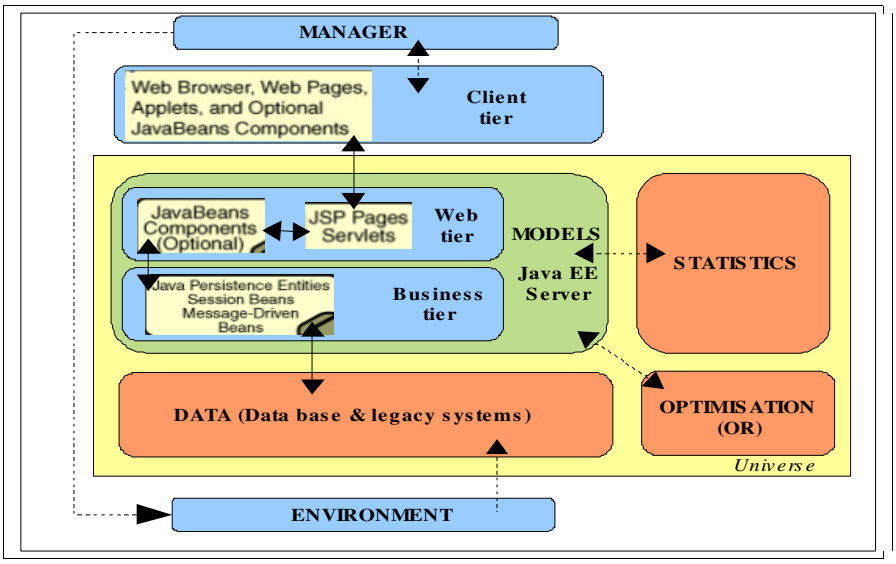
Figure 7 – Decision calculs applet integrated in a web page



The Internet is based on the client–server computing architecture. The manager using a client application the browser can access data or software applications like marketing models, statistics or other structured problem solving software located on servers anywhere in the "universe". This decoupling of software pieces and data that traditionally were

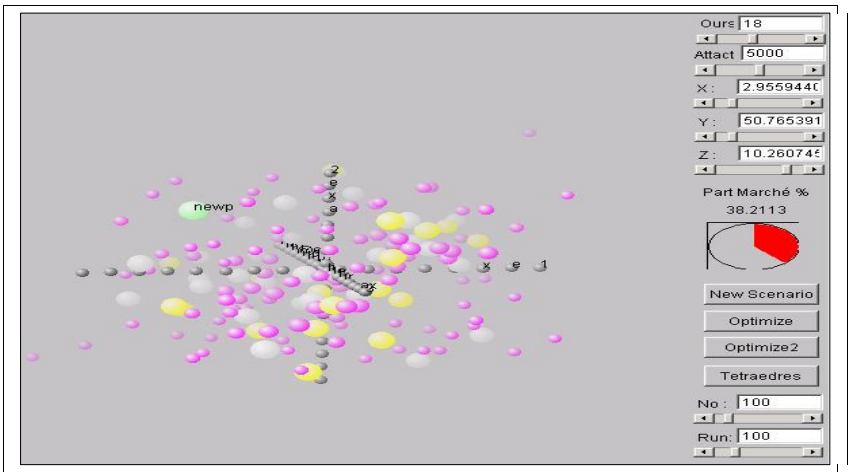
integrated and run on the same computer, brings much flexibility in the use of models. With the browser, a familiar universal tool, as as an interface, the manager can combine data and models from different sources in order to solve problems. In order to support distributed web applications the Java technology evolved progressively from browser and client side towards server side components by populating several tiers of what has evolved as a continuously growing distributed objects application framework and platform. Figure 8 tries to illustrate the composition of those tiers and to populate Little's original framework scheme.

Figure 8 – Web based Marketing decision support systems in a universe of distributed objects



The client tier is the manager's interface with the MDSS application. It uses the web browser as a "thin client"[1] reading web pages with a limited calculation capacity, relying essentially on the browser integrated scripting languages (like Javascript) and applets as client side java applications that can be embedded and are downloaded with a web page). Applets although qualified as relatively small applications can support useful decision support on the client side, as for example the applet in figure 9 that finds the optimal positioning for a band in a three dimensional perceptual space (see Calciu & Vermeersch, 2003).

Figure 9. – Applet finding the global optimum positioning of a new brand in a 3D perceptual space



1 A Java application can use a thin browser-based client or thick application client. There are trade-offs between keeping functionality on the client and close to the user (thick client) and off-loading as much functionality as possible to the server (thin client). The more functionality one off-loads to the server, the easier it is to distribute, deploy, and manage the application; however, keeping more functionality on the client can make for a better perceived user experience. Thin clients usually do not query databases, execute complex business rules, or connect to legacy applications.

The buttons and other interface objects on the right hand side of the applet are Javabeans, a special kind of classes that are self instantiating meaning that they have a constructor method (empty constructor) generating objects with default properties, and these properties can be easily read and modified with public exportable get and set methods. Javabeans can be considered components[1]. While most visual beans in this applet (figure 9) are available in the platform's libraries, the component showing the market share in the middle of the right hand side has been specially developed by us for this applet. It has been obtained by inheritance from an arc of a circle and a text label and by adding new properties.

A typical full fledged distributed application over the Internet should have at least two additional tiers a business-logic tier and a data or legacy systems-tier, that are all located on the server side of modern applications client-server dichotomy. Business or application logic is divided into components according to function, and these components are installed on different machines depending on the tier to which they belong.

Server side components which deal with our model base (see figure 8) can be organised in two tiers the web and the business tier.

The web tier components are either Servlets or pages created using JSP technology (JSP pages) technology. Servlets are Java programming language classes that dynamically process

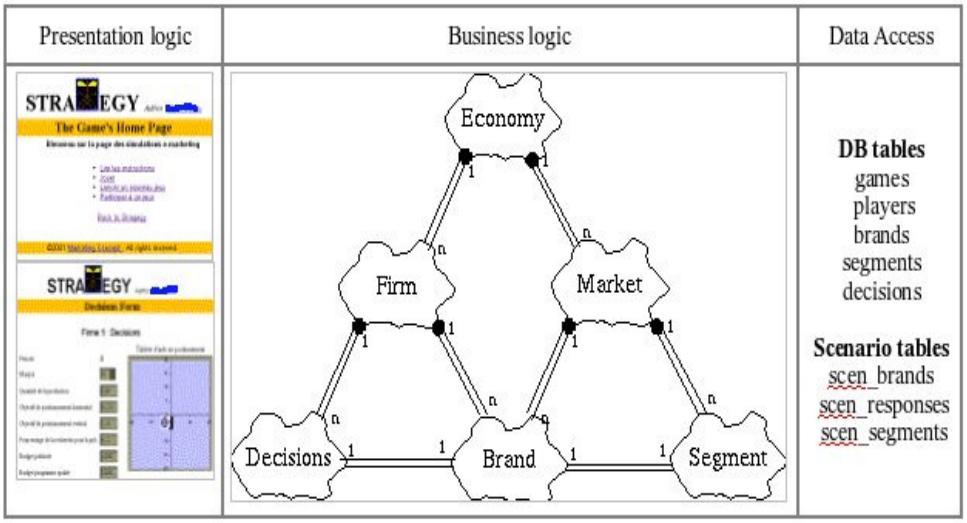
1 A component is a self-contained functional software unit that is assembled into an application and that communicates with other components.

requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.

The business tier contains code that solves or meets particular enterprise needs. It is handled by enterprise beans. Enterprise JavaBeans (EJB) are non-visual components of a distributed, transaction-oriented enterprise application. Enterprise beans are typically deployed in EJB containers and run on EJB servers. There are three types of enterprise beans: session beans, entity beans, and message-driven beans. Entity beans contain persistent data and that can be saved in various persistent data stores. Security and interoperability with other Java or non-java applications are high issues in the business tier. Business logic in a broader sense meaning application logic in contexts that are less security paranoiac can be encapsulated in non-visual Javabeans that can be connected to the servlets and JSP pages in charge with presentation logic and to database management systems.

Our prototype application uses this approach (see figure 10).

Figure 10. The Marketing Simulation as multitier e-business Application

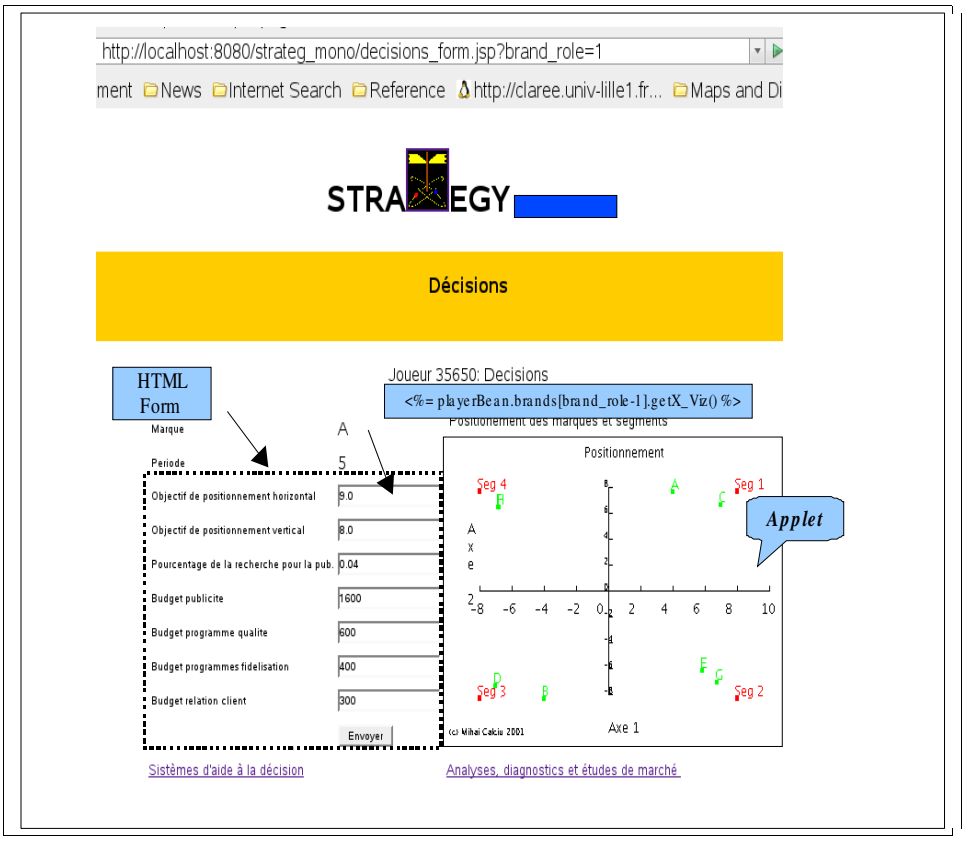


Presentation logic uses server pages (Java Server Pages - JSP) to deliver information in a visually rich, human readable form. Java Beans are used as part of a middle "business-logic" layer meant to buffer the presentation logic from the data-access.

Presentation logic dynamically produces html documents on the client side, while using server pages (JSP) and servlets on the server side that are linked to server side Javabeans that encapsulate business logic. Figure 11 shows a server page that collects a manager's (player's) decisions concerning a brand using as static elements html form and an applet. Default decisions appearing in the form are dynamically extracted from the business logic tier. In fact it is the player's (decision mak-

er's) bean brand collection that keeps using each brand beans persistent behaviour decisions taken by that decision maker in the previous period that is used as a default decision.

Figure 11– Presentation logic



Business logic in this application encapsulates at least three separate tracks, *game logic, internal model logic and persistence logic.*

Persistence logic allows beans to connect to a database through a JDBC driver. The database is managed by a database server.

It contains tables where business logic information, structured as bean properties, is persistently stored. Besides domain and application specific behaviour, Beans have also persistence behaviour enabling them to create a persistent state by inserting new records in database tables and load or store their properties that need to be persistent from or to the corresponding table records. In this application most of the beans representing Economies (or Games), Companies (or Players), Brands, Segments etc. have corresponding tables in the database, where their persistent state is kept.

The simulation is organised as a marketing game, that although minimalist, covers all important aspects of marketing. As an on-line game it is a kind of service (like banking or tourism) and can be regarded as a simple e-commerce application. The game logic is rather generic, it can be applied to several applications of this kind (Financial-, Business or Marketing games). It introduces several important topics in e-commerce applications like security, registration, authentication, session management, cookies etc. This logic is mainly encapsulated in the Game and Player Bean.

The model logic is application specific. The behaviour concerning this logic is integrated to the Brand, Segment and Market Bean. Brand beans represent the offer, segment beans express demand and integrate response behaviour and the market bean aggregates offer and demand and manages at this higher level response and transition mechanisms. The market bean integrates and embeds most of the logic from other components (beans)

Lilien and Rangaswamy (2000) suggest a simpler and pedagogical classification of models that is independent of existing distributed component technologies. They classify models according to degree of visibility and degree of integration. In figure 12 we try to illustrate how this classification applies to the models presented in this paper.

Figure 12 Marketing models classified by degree of integration and degree of visibility

Degree of Visibility	Visible Models (Interactive)	STANDALONE MODELS - Response forecasting model - Decision calculus applet - Conjoint analysis	INTEGRATED SYSTEMS OF MODELS - Marketing Simulation model
	Embedded Models ("Model Inside")	COMPONENT OBJECTS - Response beans	INTEGRATED COMPONENT OBJECTS - Market behaviour models
		Standalone	Integrated Systems
		Degree of Integration	

Source: adapted from Lilien and Rangaswamy (2000)

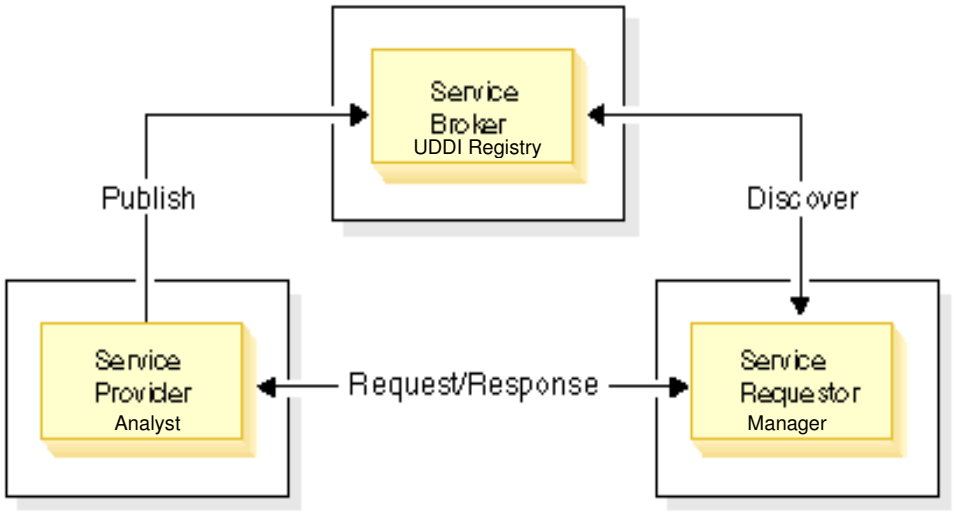
- ***Marketing models as web services***

Lilien and Rangaswamy (2000) classify models as embedded and visible ones (see figure 12) and Leeflang and Witnik (2000) speak about the need to integrate models in larger management information systems in order to stimulate their usage by managers. Probably one of the nicest ideas for freely publishing, finding and flexibly embedding models over the Internet is the web services technology.

"A Web service is a set of related application functions that can be programmatically invoked over the Internet. Business-

es can dynamically mix and match Web services to perform complex transactions with minimal programming. Web services allow buyers and sellers all over the world to discover each other, connect dynamically, and execute transactions in real time with minimal human interaction." Marketing Decision Support Models (MDSM) and Systems (MDSS) deployed as webservices and/or published in Universal Description, Discovery and Integration (UDDI) registries can easily be discovered by managers and used in a personalised way to solve problems.

Figure 13 – Accessing web services



Source: Eclipse Documentation 2006

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across

the Web. [1] Each of these self-contained services is an application that can easily integrate with other services, from the same or different companies, to create a complete business process. This interoperability allows businesses to dynamically publish, discover, and bind a range of Web services through the Internet.

The following standards play key roles in Web services: Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), Simple Object Access Protocol (SOAP). The relationship between these standards is described in Figure 14. They make web services potentially much easier to find and explore than web pages.

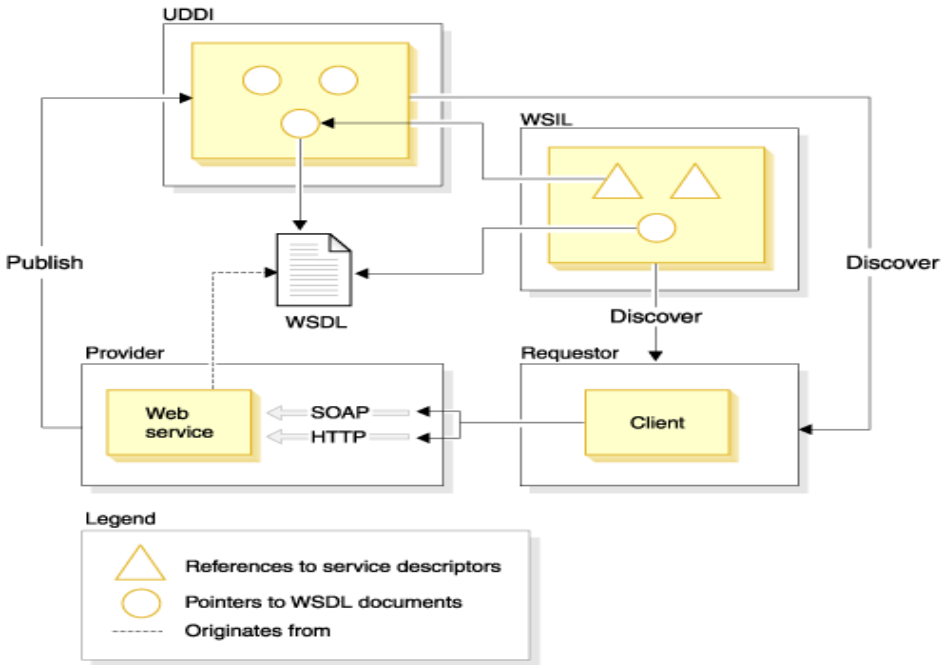
1 Web services are self-contained. On the client side, no additional software is required. A programming language with XML and HTTP client support is enough to get you started. On the server side, a Web server and servlet engine are required. The client and server can be implemented in different environments. It is possible to Web service enable an existing application without writing a single line of code.

Web services are self-describing. The client and server need to recognize only the format and content of request and response messages. The definition of the message format travels with the message; no external metadata repositories or code generation tools are required.

Web services are modular. Simple Web services can be aggregated to form more complex Web services either by using workflow techniques or by calling lower layer Web services from a Web service implementation.

Web Services are platform independent. Web services are based on a concise set of open, XML-based standards designed to promote interoperability between a Web service and clients across a variety of computing platforms and programming languages.

Figure 14. - Relationships between SOAP, UDDI, WSIL and WSDL



Source: Eclipse Documentation 2006

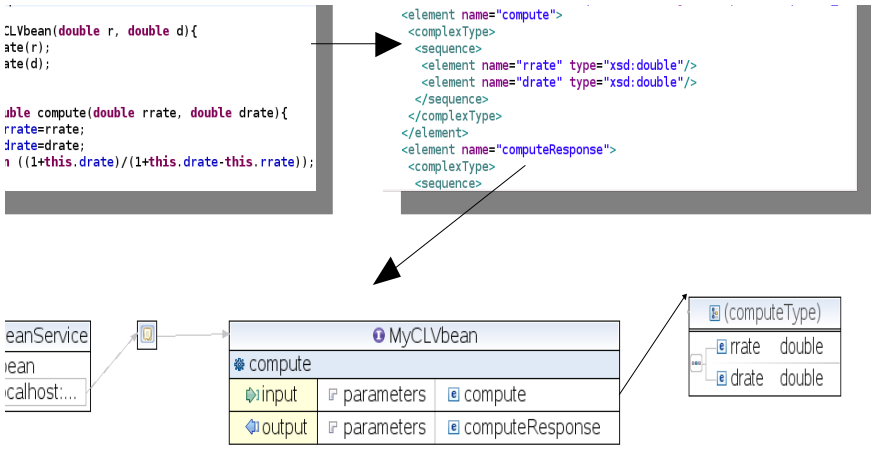
As web services have to observe complex conventions imposed by these standards, available tools assist the Web services development process. On Java platforms for example open development environments like Eclipse offer support to:

- Discover web services using Web services explorer tools that browse the UDDI Business Registries or WSIL documents to locate existing Web services for integration.

- Create or Transform WSDL documents. It is possible to create bottom-up Web services by producing their WSDL documents from existing artefacts (Java™ beans and enterprise beans). Conversely the skeletons of artefacts to support Web services can be obtained top-down from WSDL (discovered from others or created using the WSDL Editor). Let's take for example a simple Java bean that implements a method to compute unit Customer Lifetime Value (CLV)[1] that requires two arguments the retention rate and the discount rate (see figure 15).

1 the unit CLV is the lifetime value of a customer who generates one unit gain (1 euro) each year when he renews his contract. CLV calculations are used in our model in order to compute the value of the custormer base that serves as a long term performance measure for a company.

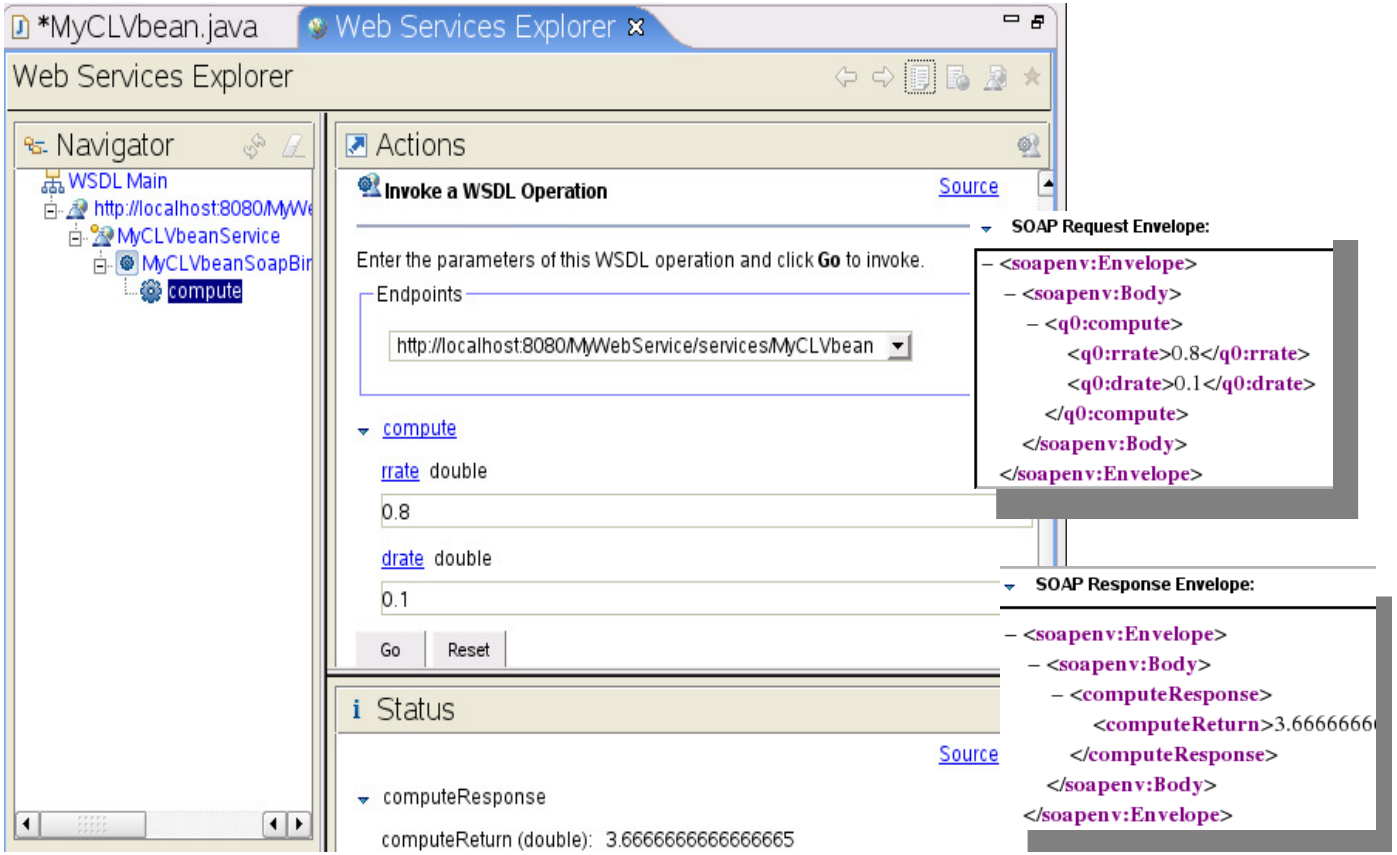
Figure 15 – Create a WebServices Description Language document from a Java bean that computes CLV



The resulting WSDL document observes an XML format for describing network services as a set of *endpoints* that operate on messages that contain either document-oriented or procedure-oriented information. The operations and messages are first described abstractly and then bound to a concrete network protocol and message format in order to define an endpoint. In figure 15 the created wsdl document is shown both in xml source and in visual form.

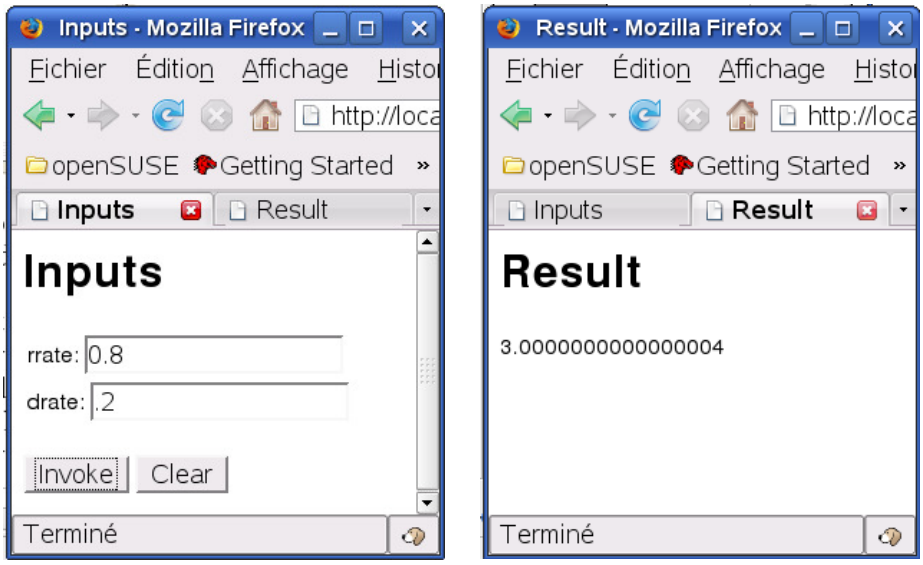
- Build SOAP accessible services as described in WSDL and clients to use them. Web services wizards assist you in generating a Java client proxy to Web services described in WSDL and in generating Java bean skeletons from WSDL.

Figure 16 - SOAP Request and Response Envelopes generated by a web services wizard



- **Deploy** Web services and their corresponding client applications into a variety of test and production environments. The previously built web service that computes CLV has been deployed on the Apache Tomcat Servlet/JSP Container. The corresponding client application consist of several Java server pages (JSP) that accept inputs from decision makers and display the results in response from the service as in figure 17.

Figure 17 - Using the service through automatically generated jsp and client applications



- Test Web services running locally or remotely in order to get instant feedback.
- Publish Web services to UDDI Business Registries, advertising the Web services so that other businesses and clients can access them. And as an alternative and complement to UDDI produce WSIL documents that

are a standardized WSDL reference. WSIL allows to go directly to the service provider and ask for the services it provides.

- ***Marketing models as documents on the semantic web***

The web was from the beginning intended to manipulate media rich documents using a mark-up language. Besides humanising the way information was accessed, it progressively shaped the evolution of DSS building technologies by bringing them closer to such documents. The large adoption of the web made it clear that it could not remain limited to search and browse static documents and that it had to deliver also dynamically generated information for business and decision support purposes.

At the beginning of the web the convenience solution to add business logic and decision support to web pages was the Common Gateway Interface (CGI) that allowed programs available on a server create web pages and publish them on the spot in response to browser intermediated client requests coming from all over the world. They could be written using the programming languages (like Perl, C etc.) available on the servers' systems who those times were almost exclusively Unix machines.

The next step to deliver web based business and decision support was the introduction of dynamic server pages technologies that mixed static html markup content with content written in a programming language specific to each server

page technology. An application server took in charge the dynamic programmatic parts of the document, transformed them into html and passed them to the web server who finally delivered a web page entirely written in html.

This mixture of static content and styling that could be found in the html part of the document with business or decision support logic represented by programming instructions was not very elegant, although more convenient than CGI where the whole web page had to be created programmatically. The creator of such a page had to be a stylist, a content composer and a programmer. Therefore some server pages solutions introduced tag libraries to hide logic (programming) behind some sort of markup tags that could be integrated with html. But this is not the ultimate solution as html is limited at displaying content only readable by humans. The ultimate solution relies upon xml which is not display oriented but content oriented and leads to a semantic web.

XML brings modelling logic much closer to documents. An XML document uses markup to identify content so that information could be easily classified and machine read. *Well formed* XML documents organise markup elements and the information they contain in a tree structure. They follow the Document Object Model (DOM) and can be *parsed* or read into memory to form a hierarchy of objects. *Serialisation* is the reverse process by which an objects' hierarchy from memory can be transformed and written into an XML document.

The analogy with object oriented programming goes further. As object types are defined by classes, mark-up and the hie-

rarchical structure to be used in an XML document can be predefined in order to produce a *valid* document. This definitions can be written in XML Scheme language or in the know obsolete Data Type Definition (DTD) language and placed in a distinct document.

Let's take for example representing products as collections of attributes as used in conjoint analysis studies. In figure 18a the XSD defines first simple elements of an attribute: level and title and then defines an attribute as a complex type containing a title and a sequence of levels and finally the collection is defined as a sequence of attributes. The XML document that respects this definition (figure 18b) presents a collection of attributes for a CDs selling web sites. The attributes are page background, image size, sound file type and text description and the levels corresponding to each attribute can be read from the document.

Figure 18 - XML document to represent products as collections of attributes with Document type definition

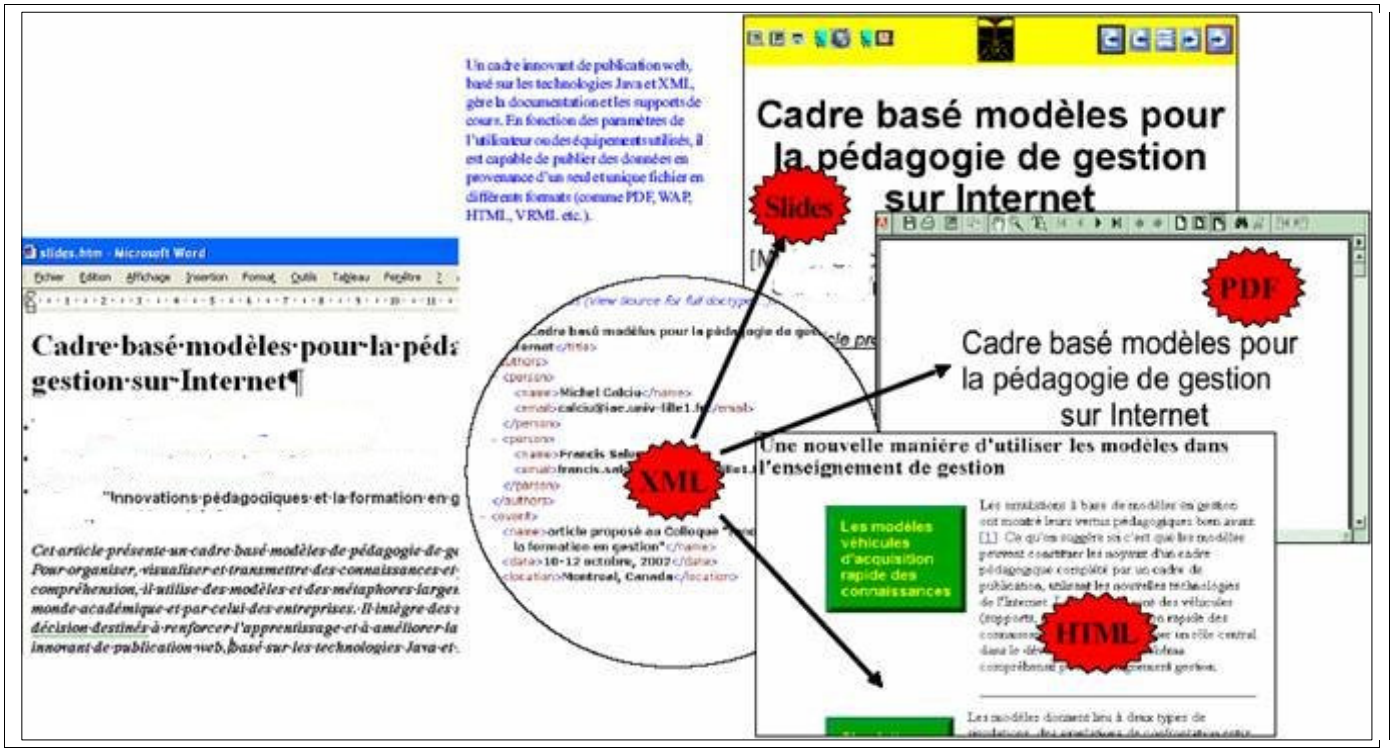
(a) XML SCHEME Document (XSD) for defining collections of product attributes	(b) Valid XML document representing the collection of attributes for a CDs selling web sites
<?xml version="1.0" encoding="UTF-8"?> <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"> <xsd:element name="level" type="xsd:string"/> <xsd:element name="title" type="xsd:string"/> <xsd:complexType name="attributeType"> <xsd:sequence> <xsd:element ref="title"/> <xsd:element ref="level" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType> <xsd:element name="attributes"> <xsd:complexType> <xsd:sequence> <xsd:element name="attribute" type="attributeType" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType> </xsd:element> </xsd:schema>	<?xml version="1.0"?> <attributes> <title>Sites web (Dreze et Zufrieden, 1997)</title> <attribute><title>Fond de page</title> <level>libre </level> <level>vert </level> <level>avec motif </level> </attribute><attribute><title>Dimension de l'image</title> <level>petite </level> <level>grande </level> </attribute><attribute><title>Fichier son</title> <level>generique </level> <level>specifique (Win, Mac) </level> </attribute><attribute><title>Description texte</title> <level>oui </level> <level>non </level> </attribute> </attributes>

As the tree structure of XML documents can be very easily read and transformed by computer programs one of the most important evolutions was the XSLT transformation language a standard way to "transform" an XML document into another XML document by associating an XSL stylesheet that contained the transformation rules. Several linked transformations where the output from one is input to the other form a pipeline.

We use such pipelined transformations to publish our courses in a somehow protected way (see http://claree.univ-lille1.fr/cocoon/mc_slides/). A chapter of a course written with a familiar word processor is saved as an XHTML docu-

ment and during publication gets transformed by a pipeline, first in an xml document as a collection of slides and then each slide that is two three paragraphs long is retransformed in html and published. In this way the chapter is only visible slide by slide, which is a convenient way to publish documents under construction as is often the case for course material. By replacing the last XSL stylesheet in the pipeline other publication formats like PDF can be obtained from the same original document as shown in figure 19.

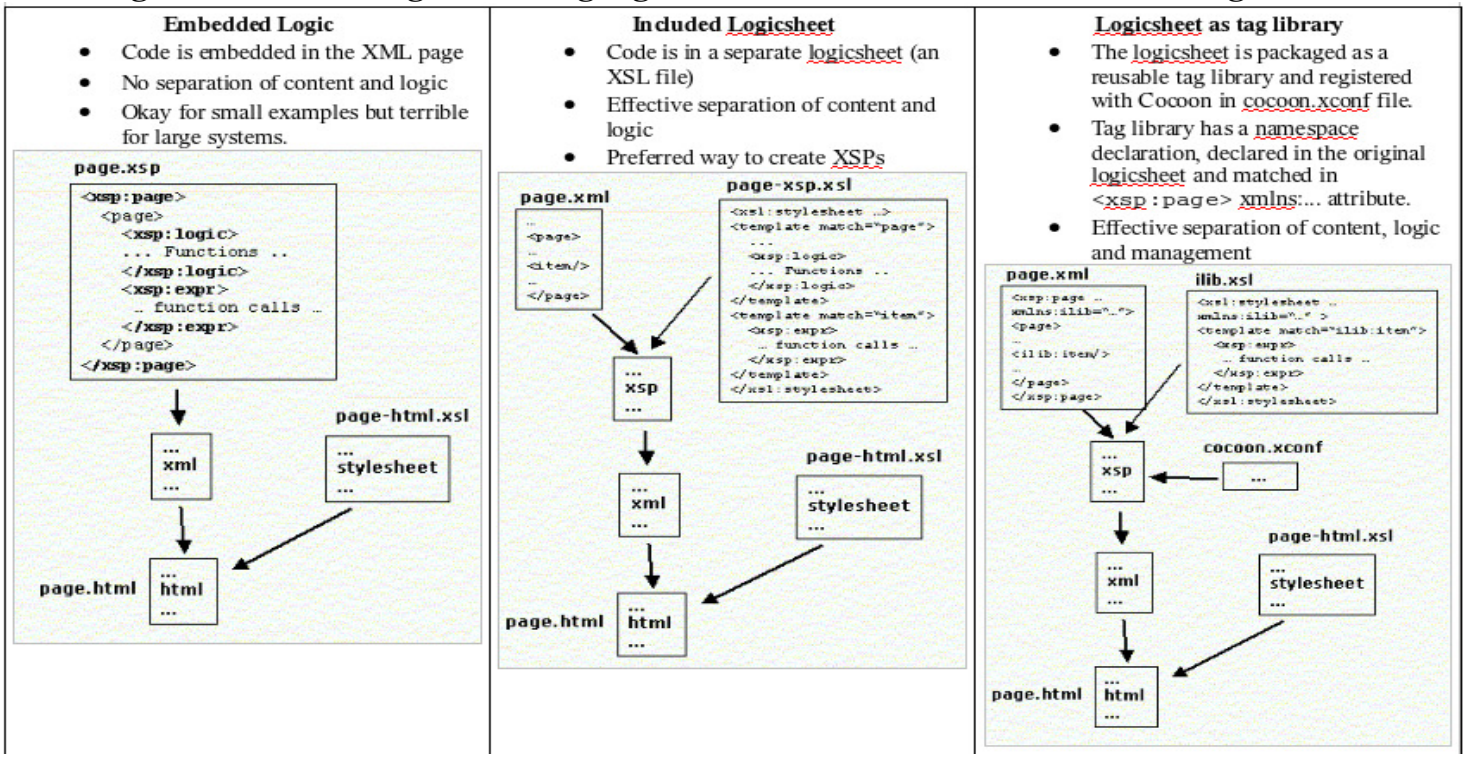
Figure 19 - Pipelined transformation of XML documents



Such transformations are seamlessly supported by an xml based application framework called cocoon (see http://cocoon.apache.org). It uses component pipelines, each component on the pipeline specializing in a particular operation. A usual pipeline uses a Generator, Transformers and a Serializer. Generators convert different data sources to XML, transformers operate XSLT transformations and serialisers generate the target document formats. This makes it possible to use a Lego(tm)-like approach in building web solutions, hooking together components into pipelines without requiring programming. Complete MDSS can be implemented as web applications using this environment.

Another aspect that promises to bring models closer to documents is related to eXtensible Server Pages (XSP) and Logicsheets. XSP are a generalisation to XML of server pages who were programming language specific (JSP – Java, ASP – Visual Basic or C etc.) and produce mainly html. As all server pages XSP allow to combine static content (which this time is XML and not HTML) with dynamic content or logic (programming). Here again in order to separate logic from content a special kind of XML transformation documents (XSL stylesheets) called Logicsheets can be used to hide dynamic content behind xml tags. Logicsheets are the XSP equivalent of taglibs. Figure 20 explains this process.

Figure 20 - Embeding and hiding logic in XSP documents and the use of Logicsheets



Source: Apache Cocoon Project Documentation

The same unit Customer Lifetime Value calculation model used before can be best integrated "as a human and machine readable document" into XSP with an included Logicsheet (see figure 21)

Figure 21 – CLV calculation as embeded logic or as markup in an XSP document

XSP document with embeded logic	XSP document with included Logicsheet	Logicsheet a XSL document with embed logic

<?xml version="1.0" encoding="UTF-8"?>
<xsp:page xmlns:xsp="http://apache.org/xsp
language="java">
<page>
<title>Computing unit Customer Lifetime Value</title>
<content>
<xsp:logic>
String srrate=request.getParameter("rrate");
String sdrate=request.getParameter("drate");
double r = Double.parseDouble(srrate); double d = Double.parseDouble(sdrate);
<para>
The unit CLV is:
<xsp:expr>(1.0 + d)/(1.0+d-r)</xsp:expr>
This result was created without using a logicsheet
</para>
</xsp:logic>
</content>
</page>
</xsp:page>
	<?xml version="1.0" encoding="UTF-8"?>
	<xsp:page xmlns:xsp="http://apache.org/xsp
	language="java">
	<xsp:logicsheet location =
	"logicsheets/mcltv.xsl"/>
	<page>
	<title>Computing unit Customer Lifetime Value</title>
	<content>
	<xsp:logic>
	String srrate=<get-rrate/>;
	String sdrate=<get-drate/>;
	</xsp:logic>
	<para>
	The unit CLV is:
	<compute>
	<rrate><xsp:expr>rrate</xsp:expr></rrate>
	<drate><xsp:expr>drate</xsp:expr></drate>
	</compute>
	This result was created by using a logicsheet
	</para>
	</content>
	</page>
		<xsl:stylesheet version="1.0"
		xmlns:xsp="http://apache.org/xsp"
		xmlns:xsp-request="http://apache.org/xsp/request/2.0"
		xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
		<xsl:template match="get-rrate">
		<xsp-request:get-parameter name="rrate" />
		</xsl:template>
		<xsl:template match="get-drate">
		<xsp-request:get-parameter name="drate" />
		</xsl:template>
		<xsl:template match="compute">
		....
		<xsp:content>
		<xsp:logic>
		String sr=<xsp:expr><xsl:value-of select="$rrate"/></xsp:expr>;
		String sd=<xsp:expr><xsl:value-of select="$drate"/></xsp:expr>;
		double r=Double.parseDouble(sr);
		double d=Double.parseDouble(sd);
		</xsp:logic>
		<xsp:expr>(1+d)/(1+d-r)</xsp:expr>
		....

The figure shows first how CLV calculation can be integrated as logic in a XSP document and then how this logic can be hidden and replaced by xml markup or tags. In this way complete separation of logic, content and styling is achieved.

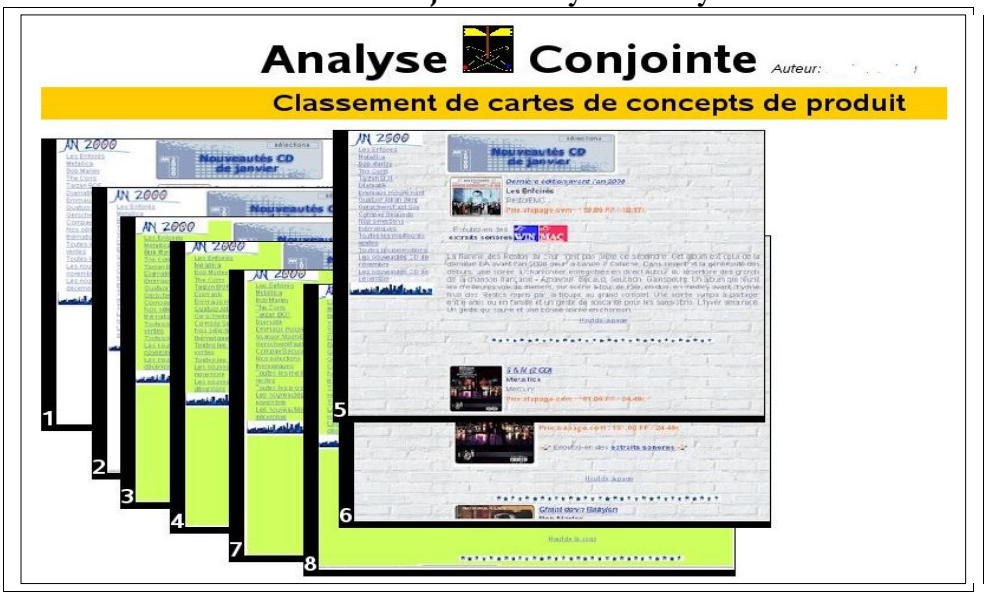
Another xml based technology that can produce logic from documents is the so called binding. It insures correspondence between data structures in xml documents and objects in memory. The program that serves to process xml data is automatically created from the document scheme, including class definitions for the memory objects. In this way the whole complexity of using files to input and output data when building applications becomes transparent. Transferring memory objects to an XML document is called marshalling, and the reverse action is called unmarshalling[1].

The publishing of a conjoint analysis study for example can be completely defined by an xml file as the one described in figure 18b. This file is then unmarshalled during the instantiation process of a Javabean in charge with this analysis. This bean controls then presentation logic by generating orthogonal experimental designs in order to display a reduced number of product concepts from the possible combinations of attribute levels described in the xml file as can be seen in figure 22 and collects the preference order given by the user for which the bean has been instantiated. The same bean computes then partworths for attribute levels, relative importance of attributes and market shares of all generated product concepts at an individual level. All this information together with data concerning the subject who took this experiment are stored as a record in a database table using the bean's persis-

1 Marshalling is for a hierarchy of memory objects what parsing is for a hierarchy of xml tags. While marshalling and unmarshalling needs valid xml documents, parsing and serialisation uses only well formed xml documents

tence logic. The resulting web application allows companies to upload or submit an xml file defining the conjoint experiment they want to launch and then indicate the appropriate web adress (http://claree.univ-lille1.fr/ConjointCastorJsp/) to all subjects she would like to see participate to the experiment.

Figure 22: - Collecting preference classification online in a xml defined conjoint analysis study



These document based technologies can take some part of the programming burden for model implementations (like for example data collection and transformation), in this way these implementations become simpler and smaller and can concentrate strictly on essential model logic.

Models or part of models can be represented as tags in a document and there from be combined and recomposed with

other models to give some solutions to complex problems. The document based programming paradigm that xml seems to have initiated can produce MDSS as complete web applications without a line of traditional programming.

▪ *Conclusions*

Marketing scientists while having good knowledge in statistics, econometrics, operations research, seem to have poor knowledge in modern programming and IT. This was not always so and is rather invalidating at least as concerns diffusion and adoption of marketing models. At the beginnings of MDSS, which somehow coincided with interactive computing, it was not rare to see MDSS implemented by the analysts themselves using some programming language (like BASIC). With the advent of micro- and personal computers and the era of office automation, spreadsheets and their "macro" programming language were often used to implement MDSS (Lilien, 1987). Paradoxically those technologies have captured marketing scientists' attention while these newer web based ones have not. Our paper tries to change this situation. It presents model building and implementation philosophies, frameworks and technologies and demonstrates solutions by using prototypical marketing models. We try to show how paradigmatic changes in systems development philosophy can affect MDSS development.

In order to facilitate understanding we present a simple but in some sense complete and rather generic marketing model. It contains essential marketing metaphors and can be used to

illustrate each technology. We suggest that this model could serve as a common denominator for some marketing scientists community who would like to learn, apply or demonstrate modern technologies and concepts affecting MDSM implementations over the Internet.

Our approach although on purpose not very sophisticated on the modelling components side is rather up-to-date as concerns IT infrastructure favouring model use. Before introducing some selected technologies we first tried to provide some guidance and criteria for choosing technologies and explained the reasons of our choice.

We also tried to match familiar model development and implementation approaches having their origin in marketing with newer, more general approaches from computer science. Some links are highlighted between decision calculus and object orientation or between a classical MDSS development/deployment scheme and some recent distributed components architectures.

We chose the Multitierd Java Web Applications environment for its portability, openness and completeness, the XML document based modelling for its elegance and possibilities to reduce programming efforts in producing applications and Web services for the flexibility they offer in locating, invoking solutions over the Internet and integrating them directly into applications. All these technologies favour diffusion of models as they become easier to implement and publish. They use a familiar and universal interface the web and strengthen it with a highly flexible back-end arsenal.

The apparent complexity and formalism defining some of these technologies like data binding or web services risks to discourage marketing scientists. What they seem to ignore is that these "complexities" can be encapsulated using available wizards through which their applications can flexibly link to data in the case of data binding or leverage the possibilities for their models to integrate, embed or be embedded over the Internet in the case of web services.

All technologies that have been described here have also been implemented on our generic marketing model and can be inspected on-line.

References

(2004). "Sprint IPMON DMS - Application Breakdown." http://ipmon.sprintlabs.com/packstat/viewresult.php?0:appsbreakdown:sj-20.0-040206.

Berners-Lee, T., R. Cailliau, N. Pellow, and A. Secret (1993), "The World-Wide Web Initiative," in *Proceedings 1993 International Networking Conference*, http://info.isoc.org/ftp/isoc/inet/inet93/papers/DBC.Berners-Lee

Bultez, Alain, (1996), Mode de diagnostics de marchés concurrentiels. *Recherche et Applications en Marketing*, 11, 4, 3-34.

Bultez, Alain, (1997), Econométrie de la compétitivité: modèles et contre-exemples. *Recherche et Applications en Marketing*, 12, 1, 21-44.

Claffy K, Miller G (1998). "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone." In "INET '98," Internet Society.

Colombo, Richard A., and Donald G. Morrison, (1989) A Brand Switching Model with implications for Marketing Strategies. *Marketing Science*, 8,1, 89-100.

Eliasberg, Jehoshua and Garry L. Lilien, Eds. (1993) *Handbook in Operational Research and Management Science, Vol.5: Marketing*, Elsevier Science Publishers B.V, Amsterdam: North Holland.

Eom S.B. (1995) Decision support systems research: reference disciplines and a cumulative tradition. *Omega Int. J. Management Sci.*, 23, 511-523.

Eom S.B. (1998) The Intellectual Development and Structure of Decision Support Systems 1991-1995). *Omega Int. J. Management Sci.*, 26, 639-657.

Festervand T.A & Harmon S.K (2001) Do marketing students need to speak XML? *Journal of Database Marketing*, 9,1,16-23

Geoffrion AM, (1987) An introduction to structured modeling. *Management Science*, 33, 5, p.547-589.

Hogue J.T adn Greco A.J. (1990) Developing Marketing Decision Support Systems Development for Services Companies. *Journal of Services Marketing,* 4,1, 21-30.

Huh, Soon-Young (1993) Modelbase Construction with Object Oriented Constructs. *Decision Sciences*, 24, 2, p.409-434.

Kazi I.H, David P Jose, Badis Ben-Hamida, Christian J Hescott et al. (2000), JaViz: A client/server Java profiling tool, *IBM Systems Journal*, 39,1, 96-117

Kernigan B.W and Richie D.M. (1978) *The C Programming Language*, Prentice-Hall: Englewood Cliffs, NJ.

Kuehn, A.A. (1961) A Model for Budgeting Advertising. *Mathematical Models and Methods in Marketing*, Bass Franck *et al.* Homewood (eds.), I11, Richard D. Irwin, 315-348.

Leeflang P.S.H & Wittink D.R. (2000) Building models for marketing decisions: past, present and future. *Intern. J. of Research in Marketing*, 17, 105-126.

Meyer-Waarden L. and Benavent C. (2001) Loyalty Programmes: Strategies and Practice. *FEDMA Research Day*, Madrid, september 14

Lang D.T. (2001) Embedding S in Other Languages and Environments, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, March 15-17, Vienna, Austria

Lang D.T. (2007) R as a Web Client – the RCurl package, Journal of Statistical Software, http://www.jstatsoft.org

Lilien. G.L. (1987) *Analyse des Décisions Marketing avec LOTUS 1-2-3* (P.Y.Desmet Trans.), Paris: Economica. (Original work published 1986)

Lilien G.L & Rangaswamy A. (2000) Modeled to bits: Decision models for the digital, networked economy, *Intern. J. of Research in Marketing*, 17, 227–235

Little, John D.C. (1970), Model and Managers: The Concept

of a Decision Calculus. *Management Science*, Vol. 16, No. 8 (April), 467-485

Little, John D.C. (1979), Decision Support Systems for Marketing Managers. *Journal of Marketing,* Vol. 43, no. 3 (Summer), 9-27.

Temple Lang D. (2001) Embedding S in Other Languages and Environments, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, March 15-17, Vienna, Austria.

Sprague, R. and E. Carlson (1982) *Building effective decision support systems*. Prentice-Hall, Englewood Cliffs, NJ.

Wierenga, B., van Bruggen, G.H., 1997. The integration of marketing problem solving modes and marketing management support systems. *Journal of Marketing* 61, 21–37, July.

Mihai CALCIU, Associate Professor, IAE, University Lille 1.

Dan SOMNEA, Professor, Ph.D., Departament of International Business and Economics, Bucharest University of Economics.