Web based technologies for model-driven decision support and simulations. Application to a dual marketing model.

Michel Calciu

Maître de Conférences

LEM – IAE - Université des Sciences et Technologies de Lille

104 av. du Peuple Belge 59043 Lille

michel.calciu@univ-lille1.fr

Web based technologies for model-driven decision support and simulations. Application to a dual marketing model.

Abstract

Based upon a dual marketing model we discuss how web based technologies help manage complexity of building decision support and simulations and accelerate their advance to the market. Some building frameworks are discussed. Several traditional web solutions are revisited, applied and compared. New ways of embedding models (models as documents, models as services etc.) are explored. 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 modellers" in order to get their models used on a larger scale.

Key words: marketing modelling, decision support, decision calculus, object orientation,internet

Introduction

Despite a long tradition in quantitative research in marketing and important accumulations of models, little diffusion of models and their implementations as decision support systems (DSS) 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). Lilien & Rangaswami (2008, p.527) refer to ‘‘the gap between realized and actual potential for the application of marketing models’’. This means that “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”.

Both DSS building and communication are significantly facilitated by new web-based information technologies. Although IT advances have shaped Marketing Decision Support Systems (MDSS) extending their use from structured, and semi-structured problems to weakly-structured ones as shown by Wierenga and van Bruggen (1997), 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 diminished 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 Web-based 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 base our demonstration and application of those technologies on a simulation and decision support system build upon a stylised 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. 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.

Traditionally DSS were built and delivered using general-purpose development software. Today many model-driven DSS are built using DSS generators [Sharda & al., 1988]. Sprague and Carlson 1982 defined a DSS generator as a computer software package that provides tools and capabilities that help a developer quickly and easily build a specific Decision Support System. They often constitute environments on which DSS are based upon. Power & Sharda

(2007) distinguish Spreadsheet based-, Web based- and Group DSS. In this paper we focus on a Web based marketing decisions support and simulation system.

A stylised marketing model

We introduce a stylised marketing simulation model combining transactional and relationship marketing, the two paradigmatic dimensions coexisting in nowadays marketing strategy. The mathematical formulation of the model is given in the appendix.

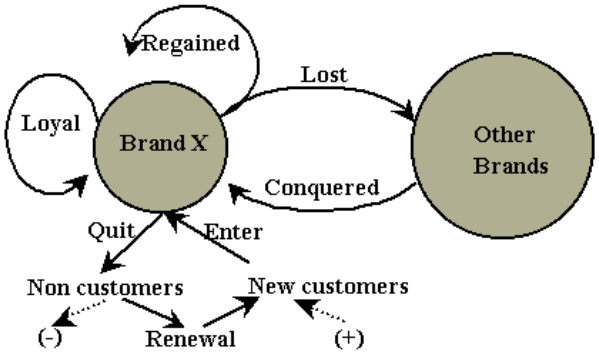
In this model, transactional marketing mix is represented by brand positioning and repositioning strategies supported by traditional communication activities. The market is a perceptual space governed by gravity laws. Brands are positioned and exert attraction upon customer segments who have mass and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

Relationship marketing mix is represented by three activities and their corresponding budget headings: interactive marketing activities (e-commerce and/or direct marketing), customer retention through quality/satisfaction programmes and loyalty programmes aimed 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 switchers" 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 customer flows from one period to the other between a given brand and the market are shown by the transition diagram in Figure 1 and are controlled at the market-segment level by a markovian process adapted from Bultez (1996, 1997)..

Figure 1. Customer flows from and to a brand



There are two kinds of repeat buyers: hard core loyal customers and regained switchers. The customers gained from other brands and those lost to other brand are also considered to be switchers. 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.

Progressive modelling philosophy, object orientation and document based programming

Before discussing technology it is useful to understand some conceptual model building frameworks that advocate flexibility and facilitate 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 satisfy in order to be used by managers. It insists mainly on model formulation aspects. Models should allow subjective parameter estimation by managers (see figure 11). Models must be "simple, robust, easy to control, adaptive, as complete as possible and easy to communicate with" (Little, 1970, p. B-466). 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.

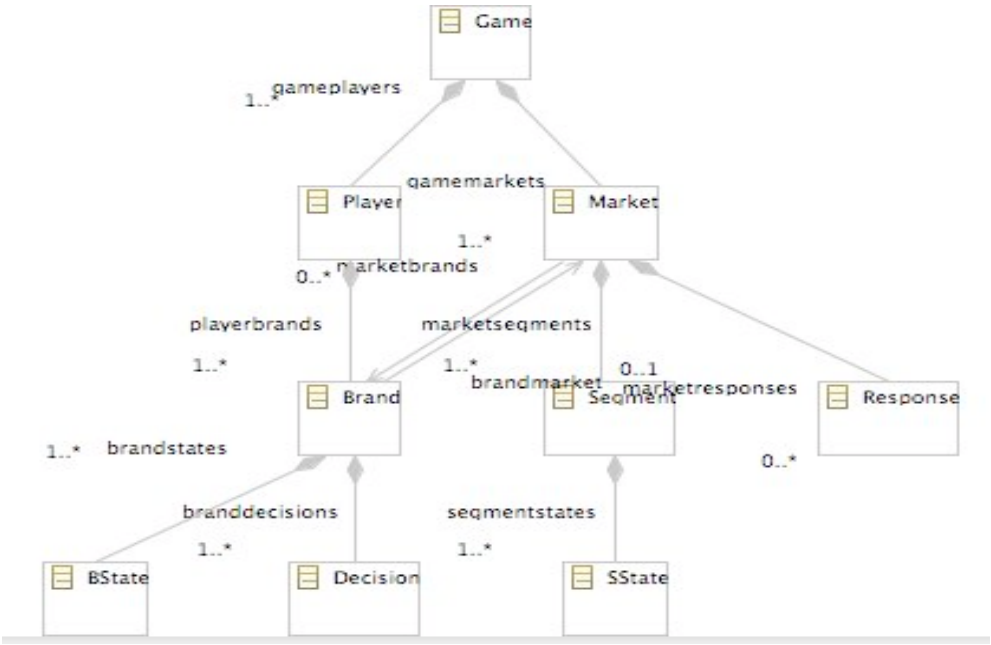
Object orientation (OO) is a way to analyse and build complex systems and decompose them into logical (classes and objects) and physical (processes module architectures) models with their static and dynamic interactions. It is largely used in computer science and became

standard on the World Wide Web. As a systems analysis tool it marked an important evolution compared to the traditional procedure oriented approaches.

Using object orientation and document based technologies to decompose the system into entity abstractions

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. In the class diagram in figure 2 we illustrate o hierarchy of containment and association relations between entity abstractions.

Figure 2 - Class diagram of a stylised marketing system



It uses the unified modelling language (UML) approach, a standard developed primarily from two formalisms for object-oriented modelling - OMT Rumbaugh et al. (1994) and Booch (1994). As can be seen from the class diagram, in this rather stylised simulation the economy (Game) contains firms (Player) and markets (Market). Firms have brands (Brand) and markets are composed of customer segments (Segments). Markets (Market) have Segments (Segment) and Response. During each simulation period firms take decisions (Decision). After each decisions round the simulation advances to the next period by calculating and adding new brand and segment states (Bstate, Sstate).

Object orientation is not only integrated in many modern computer programming languages but also in some data base management systems and even in document based approaches like

XML (eXtensible Markup Language). XML technologies bring modelling logic much closer to documents. An XML document uses markup to identify content so that information can 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 hierarchical structure to be used in an XML document can be predefined in order to produce a valid document. This can be done using the non-xml type documents called DTD (Document Type Definion) or xml type of documents called XSD (XML Schema Description). Figure 3 illustrates how the DTD defines the object structure of our simulation and of the xml document that persistently stores all information.

Figure 3 DTD (Document Type Definition)

```
<!ELEMENT Game (name,noplayers,needed,nomarkets,period,Player+,Market+)>
<!ELEMENT Player (name,password,eMail,budget,gain,period,Brand+)>
<!ELEMENT Brand (name,xNat,yNat,period, Decision+,BState+)>
.....
```

It clearly results that a simulation game contains one or more players and markets, that a player has one or more brands, that a brand has one or more decision and state objects etc. An example of an xml document defined in this way and that contains all information concerning the simulation after two periods is shown in Figure 4

Figure 4 Xml data binding the g<gamename>.xml file

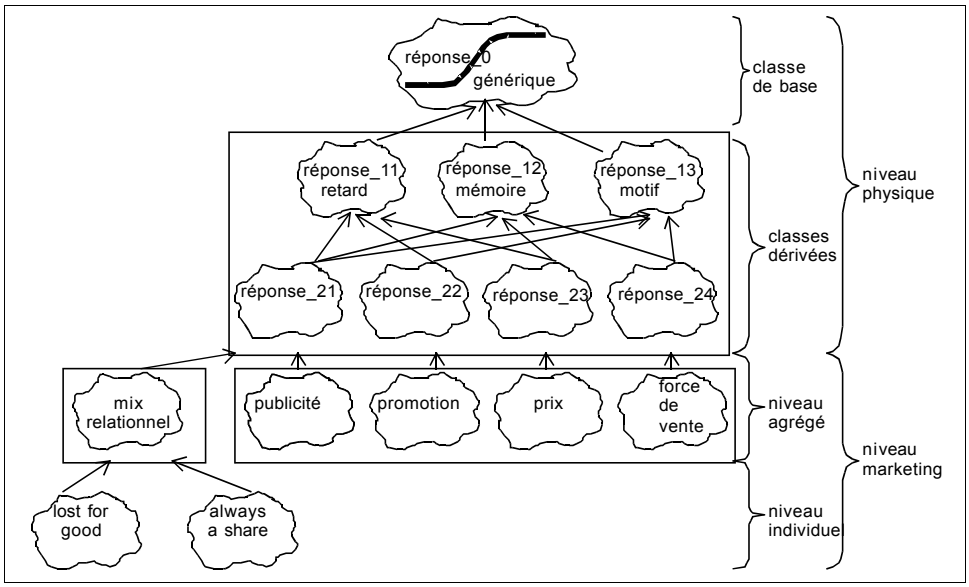
```
- <OGame name="m" needed="4" nomarkets="1" noplayers="4" period="2">
  + <OPlayer budget="1750" eMail="m1@.fr" gain="2039" name="m1" password="m1" period="1"></OPlayer>
  + <OPlayer budget="1770" eMail="m2@.fr" gain="2063" name="m2" password="m2" period="1"></OPlayer>
  + <OPlayer budget="1792" eMail="m3@.fr" gain="2089" name="m3" password="m3" period="1"></OPlayer>
  + <OPlayer budget="1814" eMail="m4@.fr" gain="2115" name="m4" password="m4" period="1"></OPlayer>
  - <OMarket brsg="0.0" noplayers="0" per="1">
    - <OSegment name="s1" period="1" sizegr02="0.04" sizegr35="0.0" sizegr6="-0.02" sizeIni="1000" valCle="1000"
      valNoncle="100" xF="14.0" xI="0.0" yF="14.0" yI="0.0">
        <OSState nCle="466" nNoncle="534" size="1000" val="520" x="4.0" y="4.0" period="0"/>
        <OSState nCle="492" nNoncle="548" size="1040" val="525" x="5.4" y="5.4" period="1"/>
      </OSegment>
    + <OSegment name="s2" period="1" sizegr02="0.04" sizegr35="0.0" sizegr6="-0.02" sizeIni="1000" valCle="1000"
      valNoncle="100" xF="14.0" xI="0.0" yF="-14.0" yI="0.0"></OSegment>
    + <OSegment name="s3" period="1" sizegr02="0.04" sizegr35="0.0" sizegr6="-0.02" sizeIni="1000" valCle="1000"
      valNoncle="100" xF="-14.0" xI="0.0" yF="-14.0" yI="0.0"></OSegment>
    + <OSegment name="s4" period="1" sizegr02="0.04" sizegr35="0.0" sizegr6="-0.02" sizeIni="1000" valCle="1000"
      valNoncle="100" xF="-14.0" xI="0.0" yF="14.0" yI="0.0"></OSegment>
      <OResponse a="1.5" b="1.1" max="1.75" min="0.5" mixRole="1" result="0.501809" type="1"/>
      <OResponse a="3.63636" b="3.7" max="3.0" min="0.45" mixRole="1" result="0.45" type="1"/>
```

In another paper (Calciu & Popa, 2010) we suggests a document-based framework for building marketing decision support and simulation systems. This framework transfers most of the marketing model implementation work to documents. It replaces many computer language programming tasks by human readable documents leading to increased separation of concerns and acceleration of model access to the market. The framework uses what we call document based programming. It essentially applies recent xml document technologies, like xslt transformations, xml data- binding, xml-schemas to replace programming tasks by standard automated processes. Some of these documents, by generating a combination of UML diagrams specify a complete model of an application. Using appropriate tools that often are regrouped in a modelling framework[1] part of or, in simple cases, all of an application can be generated.

Modelling customer response as an object oriented hierarchy of abstractions

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

1 We used the Eclipse Modelling Framework.

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").

Our progressive response based modelling approach is particularly fit for a category of models called aggregate response models. An aggregate response model "seeks to relate sales, share, distribution, or other criterion variables directly to the marketing actions involved" (Little, 1975 p.633). In this category of models one can find marketing decision support models like CALLPLAN (Lodish, 1971), BRANDAID (Little, 1975), STRATPORT (Laréché & Srinivasan, 1981,1982) or SCAN*PRO (Wittink et al., 1988) but also model based marketing simulations like Markstrat (Laréché & Gatignon, 1990) and the dual marketing model presented in this paper.

Deploying MDSS over the Web

Definitions and early arhitectures

In 1966, Kotler introduced the concept of a ''MarketingNerveCentre'', providing marketing managers with ''computer programs which will enhance their power to make decisions.'' (Wierenga & al., 2008, p. 561)

The concept of marketing decision support systems (MDSS) has been introduced by Little (1979) and defined as a ''coordinated collection of data, systems, tools and techniques with supporting software and hardware by which an organization gathers and interprets relevant information from business and environment and turns it into an environment for marketing action'' (p. 11).

Wierenga & Van Bruggen (1997) used the term ''marketing management support systems'' for systems additionally supporting marketing decision-making in weakly-structured areas that are primarily qualitative, and knowledge-driven.

In a model-driven decision support system quantitative models are the dominant components that provide the primary functionality. They differ from communications-driven, data-driven, document- driven and knowledge-driven DSS (Power, 2002). They allow to manipulate model parameters in order to examine outputs in more (sensitivity analysis) or less (ad hoc "what if?" analysis) systematic way. Model-driven DSS differ from the computer support used for a decision analytic or operations research special decision study at least in two points: first a model in a model-driven DSS is made accessible to a non-technical specialists and second it is intended for some repeated use in the same or a similar decision situation (Power & Sharda,

2007).

Little (1979) suggested one of the best known MDSS frameworks, 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. While the model component solves semistructured marketing problems it can access the statistics and optimisation component in order to solve structured problems.

Web-based architecture for marketing decision support and simulation systems

Web-Based DSS advantages and disadvantages

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 (Berners-Lee, 1993). “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.

Some advantages web based DSS are: ease of use; collaboration facilities; flexible licence and deployment modes; model reuse; cross platform capabilities; controlled access; wide availability; versioning, customisation and maintenance; integration and operability

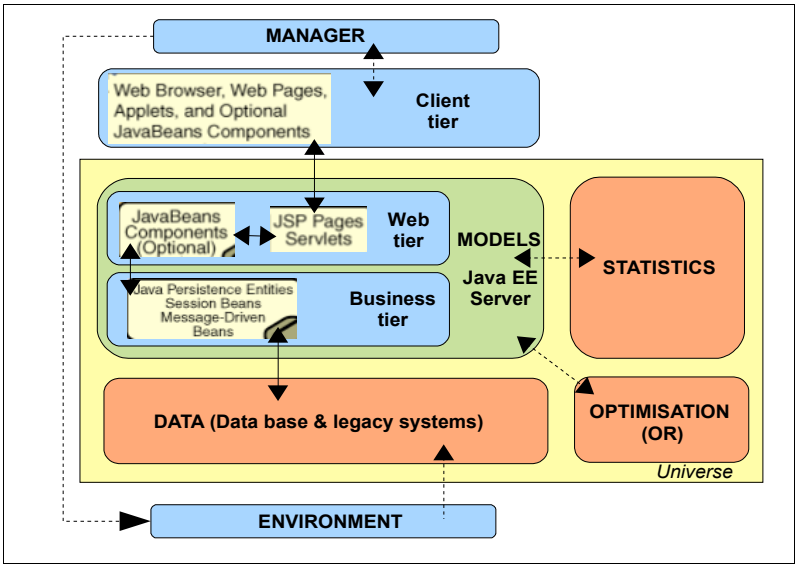
Eventual disadvantages can be: loss in speed; graphical user interface limitations; security vulnerability; stability potentially affected by disappearance of a site hosting a distributed component; licensing restrictions. For more details concerning these advantages and disadvantages in the field of web based simulations the reader could refer to Byrne & al. (2010).

Web-based component structure

The web based component structure we suggest in this paper adapts Little's (1979) historical

MDSS framework by integrating web-based component technologies within that framework's interface and model-base component. These technologies that are listed in figure 6 will be more thoroughly discussed later in this document. The other components data, statistics and optimisation refer to specialised software categories that help solve structured problems issued by the model component when dealing with a semistructured problem. The integration of web based technologies within those specialised software categories is well documented and will only marginally be dealt with here.

Figure 6 – Web based Marketing decision support systems in a universe of distributed objects



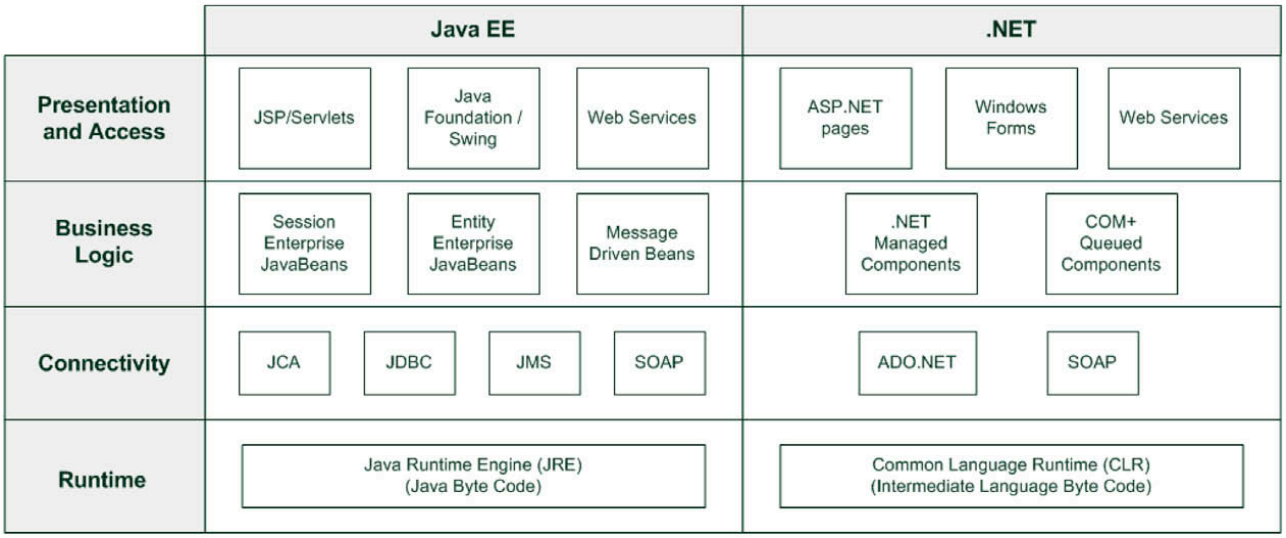
While there is a certain convergence in the evolution of web based applications development and implementation platforms, the technologies used by them are rather different and complex. Therefore adopting such a platform is a long term choice. A modern web based architecture needs to be component based and distributed. Components are the natural extension of objects (see object orientation) and are particularly well suited for distributed computing.

Syzyperski (2003) defines a software component having to be a unit of deployment and independent of one another and communicate only by defined means. Bertoa et al. (2006) state that the component-based software development approach tries to improve the flexibility, reusability and maintainability of applications, helping develop complex and distributed applications deployed on a wide range of platforms, by plugging prefabricated software components, rather than building these applications from scratch. Although distributed applications preceded the Web, the Web is now influencing distributed applications. As computing environments are becoming more distributed and heterogeneous,

middleware technologies are emerging that offer standard infrastructure to deal with the interoperability of such applications.

A the time of this writing there are two main platflorms for building and delivering Web-based application in general and DSS in particular Microsoft's COM+ and Sun's EJB both are considered as transactional component middleware (Britton & Bye, 2004). The key components of Sun's Java Enterprise Edition and Microsoft's .NET web development platforms are presented in figure 7.

Figure 7 Java EE and .NET Architectures



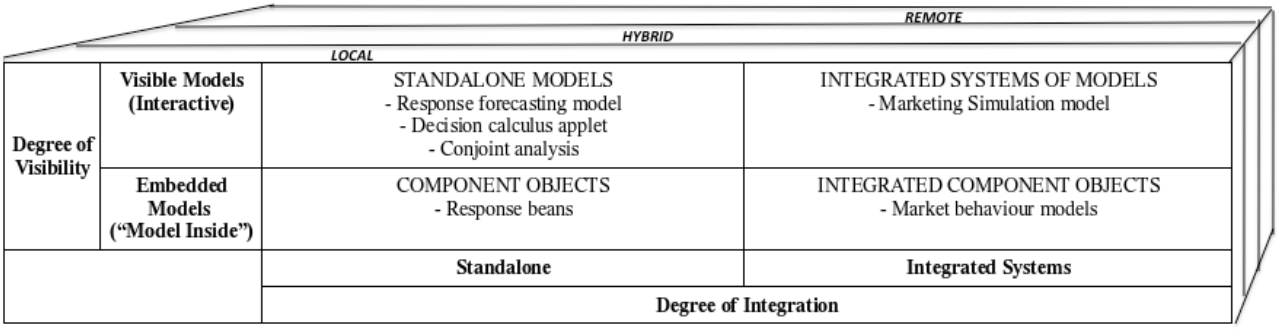
Source: Byrne & al (2010) adapted from (Kachru & Gehringer, 2004)

Our choice of the Java based platform is guided by the need to favour MDSS diffusion and our criteria are openness, availability, portability and recognition by the computer science profession. 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). 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.

Lilien and Rangaswamy (2000) suggest a classification of MDSS (marketing engineering) models that can be deployed on the World Wide Web and accessed over the Internet. To the original two criteria degree of integration and degree of visibility we add the so called client-server dimension. We distinguish between local, remote and hybrid MDSS. When both the visual (interactive) logic and the model logic are concentrated on the client computer the application is local. When both are on the server it is "remote" and when the visual logic is on the client and the model logic is totally or partly on the server it is hybrid. In figure 8 we try to illustrate how this classification applies to the models presented in this paper.

Figure 8 Marketing models classified by degree of integration and degree of visibility



Source: adapted from Lilien and Rangaswamy (2000,2008)

As to this classification the model based marketing simulation we have presented is an integrated system of models as it combines visually the core simulation with a series of MDSS that can also function as standalone models. Such visual standalone applications are a response forecasting model, a decision calculus applet, a conjoint analysis and an optimal positioning system that are implented as local or hybrid applications and will be presented later. Embedded models are for example the java components that define segments' response behaviour whose object oriented complexification approach has been discussed before. An integrated embedded model is the markovian transition model that governs market behaviour and controls customer flows from and within each positional segment depending on key and non-key customers' response models.

As -ceteris paribus- visible models are more complex than their embedded counterparts, as they add to the hidden core behaviour interactive and visual elements, the remaining part of the paper will discuss web based technologies applied to visible local, hybrid and remote MDSS.

Web based technologies applied to visible local, hybrid and remote MDSS

Hybrid models

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 de-coupling 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 7 tries to illustrate the composition of those tiers and to populate Little's original framework scheme.

A typical hybrid MDSS has its visual interface on the client side and the hidden model logic on the server side. Our simulation is such a hybrid application. The client tier is the manager's interface with the MDSS application. It uses the web browser as a "thin client"[2] 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).

2 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.

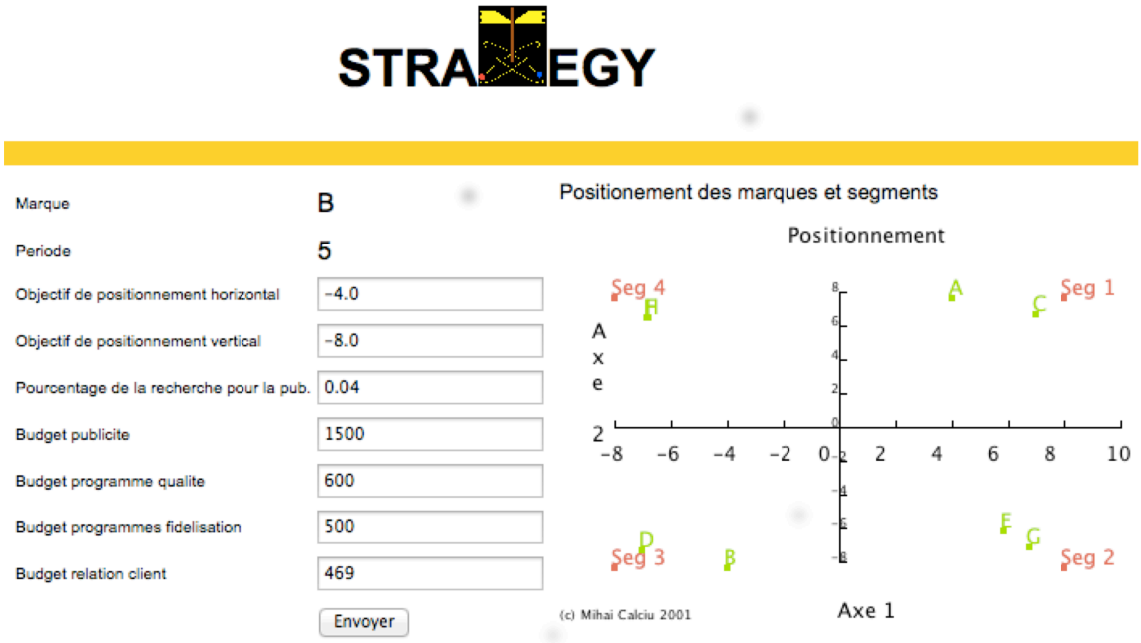


Figure 16 shows the simulation's main web interfaces that contains an HTML Form where the decision maker fixes positioning objectives for his brands and allocates budget to transactional and relationship mix activities. In this case the applet is a java application produces a visual perceptual map representation of the market in which actual brand positions are confronted to segments' ideal points.

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.

Server side components which deal with our model base (see figure 7) can be organised in two tiers the web and the business tier. The web tier components are either Servlets or dynamic server pages created using JSP (Java Server Pages) technology. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content. In this case they are in charge with what is called the *Presentation logic* and deliver information in a visually rich, human readable form.

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*[3]. Enterprise beans are typically deployed in EJB containers and run on EJB servers. Our simulation's core behaviour is encapsulated in non-visual Javabeans derived from the hierarchy of entity abstractions that form the class diagram in figure 2. They form a middle "business-logic" layer meant to buffer the presentation logic from the data-access. *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. The *game logic* is rather generic, it introduces several important topics 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)

Local web based MDSS

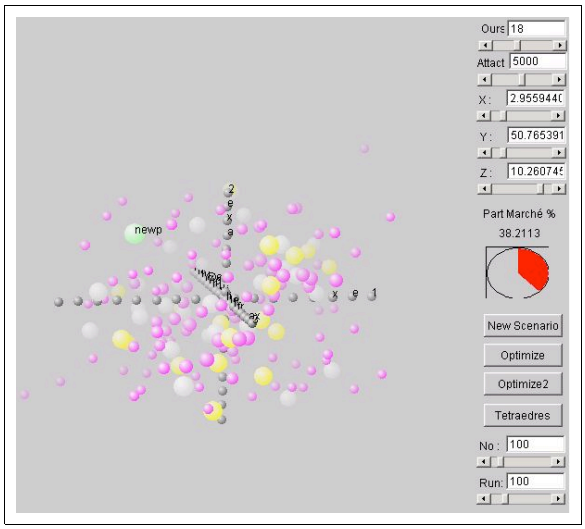
In local web based MDSS both the visual and the model logic are downloaded seamlessly by the client (web browser) to the user's local computer. The responsibility for execution shifts completely from the server to the client. The server simply becomes a central distribution point for the simulation without performing any real work.

Historically it was common to use Java applets in such situations. They can be embedded in web pages and use computing power on the client-side from a virtual computer the Java Virtual Machine (JVM) that was traditionally linked to the web browser.

3 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.

Applets although qualified as relatively small applications can support useful decision support on the client side, as for example the applet in figure 10 that finds the optimal positioning for a brand in a three dimensional perceptual space (see Calciu and Vermeersch, 2003).

Figure 10. – Applet finding the global optimum positioning of a new brand in a 3D perceptual space



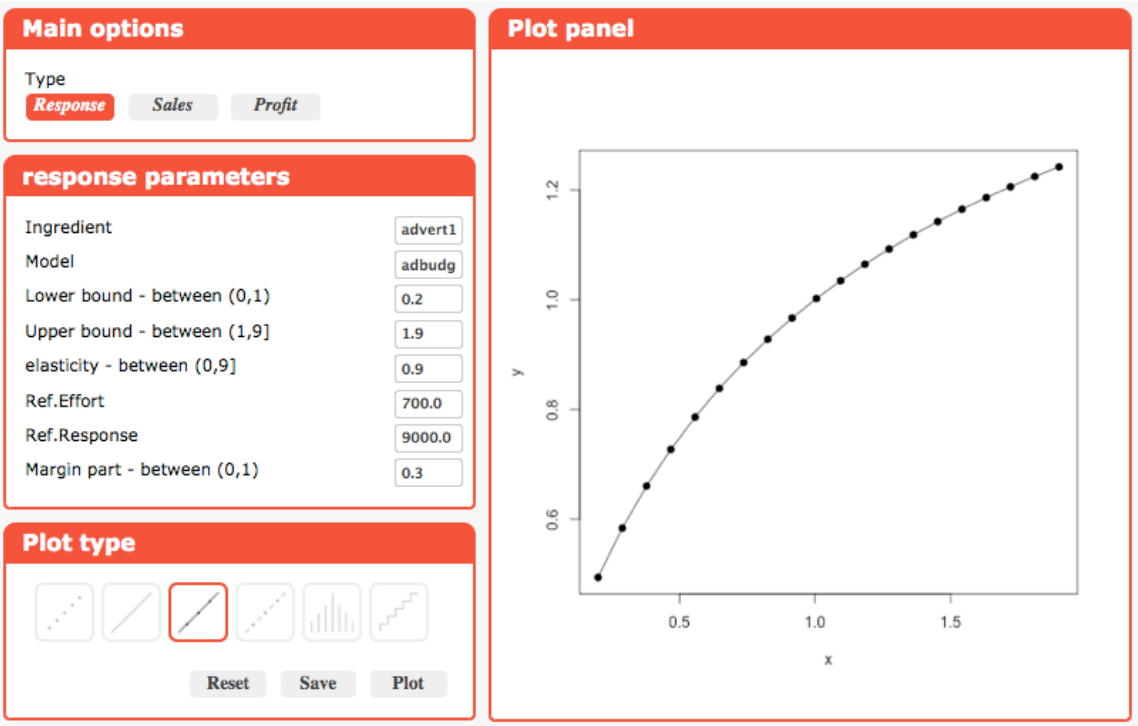
The buttons and other interface objects on the right hand side of the applet are Javabeans, a special kind of classes that are self instantiating meaning that they have a constructor method (empty constructor) generating objects with default properties. Javabeans are components[4]. While most visual components in this applet are provided by a java graphical user interaction (GUI) library, 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 multiple inheritance from classes like arc of a circle and text label. At the beginning when the web was used as hypertext medium and was not very good for software applications, applets offered the first professional GUI[5]. This has fundamentally changed with the adoption of AJAX (Asynchronous JavaScript + XML) and the continuous improvement of graphical capabilities of web browsers that crystallise around the emerging HTML 5 standard. As a result, glamorous user interaction facilities can be implemented using open-source Javascript libraries (jQuery, ExtJS). We illustrate this with a tool, we have developed, that gives decision

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

5 Although applets tend to become obsolete in web-pages, and the applet tag has been omitted in HTML5, applets remain the only tool with which one can do "*anything*" in client-side web-applications, while newer HTML5 and AJAX technologies can only do "*many things*"

makers the opportunity to visually indicate market response to marketing efforts by subjective estimation or decision calculus as in figure 11 and with a triple view as a relative response index, as sales or as profit.

Figure 11 – Local decision calculus application using AJAX with jQuery



The page layout is dynamically generated in html using div layers with appropriate id and class attributes in order to be easily identified by jQuery's powerful DOM querying expressions. As long as none of triple view options is selected only the main options panel and an empty plot panel are visible the other panels are hidden by jQuery commands. When one option is selected the other panels become visible and the user can change response parameters and select a plot type. Both local applications presented in this section are standalone MDSS.

Remote models

In remote MDSS, both the visual and model logic are located and execute remotely, on the server-side. Access to these is through a browser which is a thin client. This is similar to a job request on a batch processing system. Parameters are submitted to the simulation engine through the Web server, and results are returned to the user once the simulation has finished running. The visual results generated by the server can be transmitted to the graphical interface in the browser using, for example, Common Gateway Interfaces (CGIs), sockets,

Java remote method invocation (RMI), JavaBeans, Common Object Request Broker Architecture (CORBA), remote procedure call (RPC) or via front-end applications.

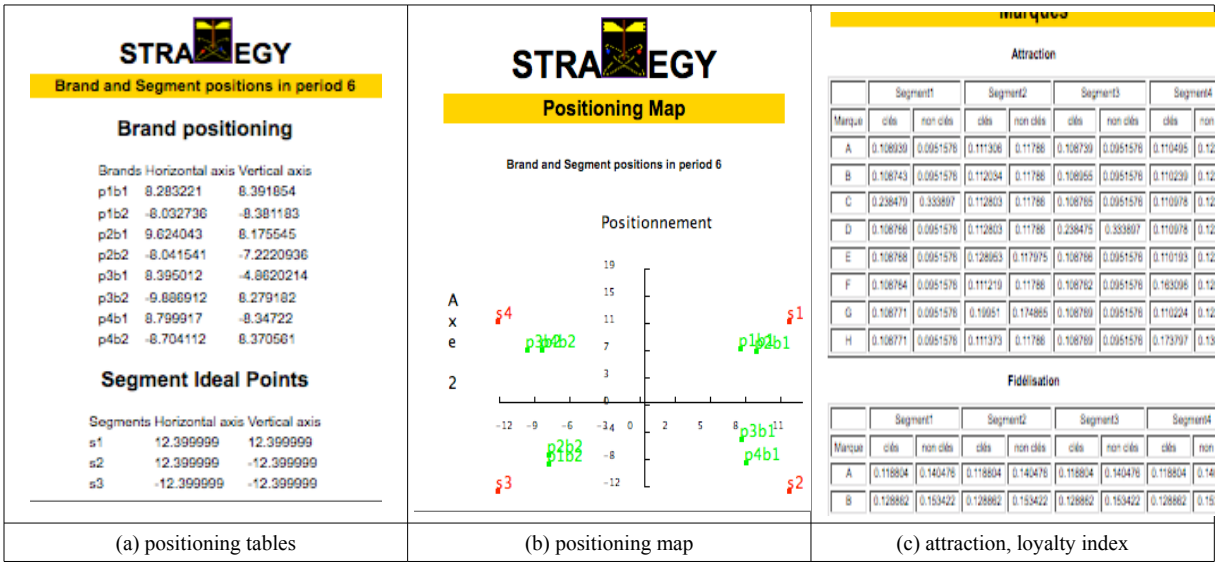
We have developed a batch version of our marketing simulation that uses two XML technologies: XML data-binding and XSLT document transformations in order to prepare the visual logic of the application on the server-side.

XML data binding refers to the process of representing the information in an XML document as an object in computer memory and vice-versa. An XML data binder accomplishes this by automatically creating a mapping between elements of the XML schema of the document we wish to bind and members of a class to be represented in memory. 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[6] . The XML schema of our simulation is automatically generated from DTD document presented in figure 2 it prepares the xml binding process which allows the application to read (unmarshall) uploaded xml files containing players' decisions and after processing a simulation period to output (marshall) simulation results as xml files as the one illustrated in figure 4.

The visual part of the application is prepared using one of the most exciting XML technologies the Extensible Stylesheet Language Transformation (XSLT). It is a standard way to "transform" an XML document into another XML document by associating an XSL style-sheet that contains the transformation rules. In our simulation by associating various XSL style-sheets located on the remote server the xml file presented in figure .. is transformed into web pages (xhtml files) that display on the browser media rich information extracted from them. Decision makers will be able to see information concerning firms, brands, segments and customer flows in visual and/or tabular form. Some examples are shown in figure 12.

6 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

Figure 12 information extracted by xsl stylesheets from a game's xml file



(a) positioning tables | (b) positioning map | (c) attraction, loyalty index

An advantage remote applications is that larger MDSS can run on powerful, high-end computers and analysts or decision makers can get access to the results with a browser. Web services are a well adapted technology to invoke such applications programmatically over the Internet. Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. [7] 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 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). Marketing Decision Support Models (MDSM) and Systems

7 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.

(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.

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, Calciu & Benavent, 1995). 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 Multitiered 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 visited on-line.

References

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

Alter, S.L. (1980) *Decision Support Systems: Current Practice and Continuing Challenge*, Addison-Wesley, Reading, MA.

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

Bertoa, M.F., Troya, J.M. & Vallecillo, A. (2006), Measuring the usability of software components, *Journal of Systems and Software,* 79, 3, 427–439.

Booch G. (1999) Object-oriented Analysis and Design with Applications (2nd edition). Reading, MA, Addison-Wesley

Britton, C. & Bye, P. (2004), *IT Architectures and Middleware*, second ed., Addison-Wesley, Boston.

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.

Byrne, J., Heavey, C. & Byrne, P.J. (2010), A review of Web-based simulation and supporting tools *Simulation Modelling Practice and Theory*, 18, 253–276

Calciu M. et Popa I. (2010) "A document based marketing decision support and simulation framework. Application to a customer attraction/retention model.", 9-th, International Conference Marketing Trends, Venice, January 21-23

Calciu M. et Vermeersch G. (2003) "Optimal continuous location in the geographic and perceptual space using attractiveness and market share", INFORMS Annual Meeting (Spatial Marketing Session), Atlanta, October 19-22

Calciu M., Benavent C. (1995) " Système d`estimation des taux de réponse à des opérations promotionnelles avec apprentissage catégoriel et longitudinal ", Journées de recherche de l`IAE de Lille, September 27-28

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.

Kachru, S. & Gehringer, E.F. (2004), A comparison of J2EE and .NET as platforms for teaching web services, in: *34th ASEE/IEEE Frontiers in Education Conference (FIE)*, Savannah, GA, USA, 2004, pp. 12–17.

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.

Kotler, Ph. (1966), A Design for the Firm's Marketing Nerve Center. *Business Horizons*, 9, 3, 63–74

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.

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

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.

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.

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

Power,D.J., (2000), Decision Support Systems Hyperbook, http:// dssresources.com/dssbook/ (Fall).

Power,D.J. (2002), *Decision Support Systems: Concepts and Resources for Managers*, Greenwood/Quorum Books, West- port, CT, 2002.

Power, D.J. (2004), Specifying an expanded framework for classify- ing and describing decision support systems, *Communica tions of the Association for Information Systems,* 13 (13) ,158 – 166

Power,D.J., Sharda, R. (2007), Model-driven decision support systems: Concepts and research directions, *Decision Support Systems,* 43, 1044–1061 .

Rumbaugh, J., Blaha, M., Premerlain, W., E.F., Lorenzo W. (1994), *Object-oriented modeling and design*, Prentice-Hall, Engle- wood Cliffs, N.J.

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.

Sharda, R., Barr, S., McDonnell, J. (1988), Decision support systems effectiveness: a review and an empirical test, Management Science, 34,2,139 – 159.

Syzyperski, C. (2003), Component technology: what, where, and how? in: *Proceedings of the 25th International Conference on Software Engineering*, IEEE Computer Society, Portland, Oregan, 2003, 684–693.

Sprague, R.H.,Jr., (1980), *A framework for the development of decision support systems*, Management Information Systems Quarterly, 4, 4,1 – 26.

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.