

Title	Decision Support Systems for Mouse-clickers
Author(s)	Chunhui, XU
Citation	
Issue Date	2005-11
Type	Conference Paper
Text version	publisher
URL	<a href="http://hdl.handle.net/10119/3881">http://hdl.handle.net/10119/3881</a>
Rights	2005 JAIST Press
Description	The original publication is available at JAIST Press <a href="http://www.jaist.ac.jp/library/jaist-press/index.html">http://www.jaist.ac.jp/library/jaist-press/index.html</a> , IFSR 2005 : Proceedings of the First World Congress of the International Federation for Systems Research : The New Roles of Systems Sciences For a Knowledge-based Society : Nov. 14-17, 2001, Kobe, Japan, Symposium 3, Session 3 : Intelligent Information Technology and Applications Networks and Agents

# Decision Support Systems for Mouse-clickers

Chunhui XU

Dept. of Management Information Science  
Chiba Institute of Technology  
Chiba 275-0016, Japan

## ABSTRACT

This paper proposes the notion of decision support system (DSS) for mouse-clickers, a DSS which does not require its users to have any professional knowledge in decision science and computers, and suggests a new architecture for these kinds of supporting systems. Such a DSS for investment is developed to illustrate the validity of such an architecture.

**Keywords:** Decision support system, Modeling, Knowledge requirement, Mouse-clickers, Model generator.

## 1. INTRODUCTION

People believe that computers can do something to help them in making complicated decisions, this belief initiated research on decision support systems in 1960s, and various DSSs had been developed since then, see [1] for a brief history of DSS, and [2] for a view of the field of decision support systems.

However, few people went to DSS when faced with decision problems even if some DSSs are available, because people lost their interests in using DSSs before getting through the fat instruction books, let alone those lacking in knowledge required to understand them.

This paper believes that DSSs can benefit a wider range of people when the knowledge requirements for the users are reduced to certain level, and it is practically worthwhile to develop DSSs for non-professionals in decision science and computers, called mouse-clickers in this paper. This study is done under such this belief.

The contents are arranged as follows: Section 2 analyzes the structure of conventional DSSs and its problems from the users' position, Section 3 proposes a new architecture for DSSs which suits mouse-clickers' book. In order to show that the proposed architecture is practically appropriate and implementable, Section 4 develops an investment support system which does not require its users to do any modeling and programming.

## 2. CONVENTIONAL DSSs and THEIR PROBLEMS

DSSs are computer-based systems that aid users in judgment and choice activities. While a variety of DSSs exists, the following three fundamental components can be found in many DSSs.

- Database management system (DBMS).

A DBMS stores data that are relevant to the class of problems for which the DSS has been designed and provides logical data structures with which the users interact.

- Model-base management system (MBMS).

A MBMS assists users in model building, modifying, storing and solving.

- User Interface (UI).

UI enhances the ability of the user to utilize and benefit from the DSS. It aids users in interaction

Figure 1 shows the architecture of the conventional DSSs. Because the UI and MBMS in most DSSs are not in a high-quality finished form, their users need to interact with the models directly for using DSSs correctly, such as modifying or customizing the models. Some DSSs even require the users to develop solvers to solve the models they built. These requirements made DSS beyond the reach of most non-professionals in modeling and programming.

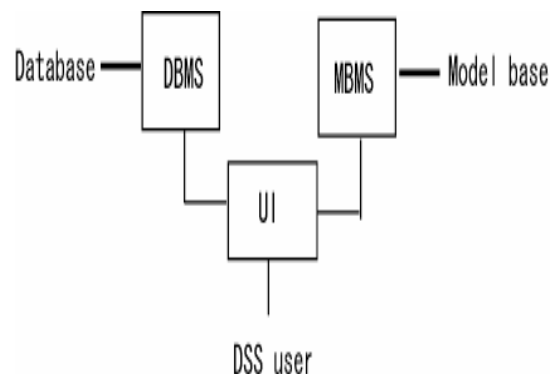


Figure1. Components of conventional DSSs

This situation will change if DSSs can generate models according to the information their users may

provide, and get solvers ready for the models built.

Next section will propose an architecture for such DSSs, which do not require the users to have any knowledge of building and solving models, all technical processing for making a better decision are done by the system instead of the users.

### 3. A NEW ARCHITECTURE for DSSs

The architecture we suggest is shown in Figure 2 below.

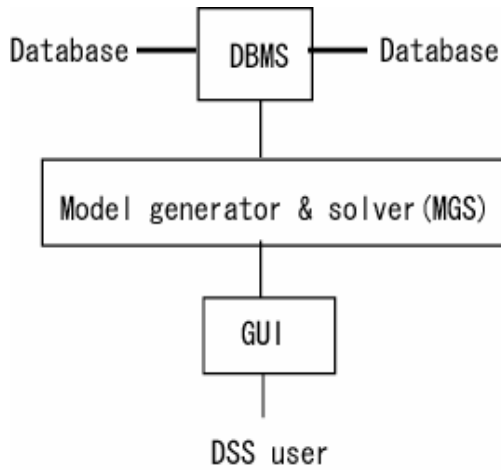


Figure 2. A new architecture for DSSs

In this architecture, we add a model generating function to the system and emerge it with the model management system to form a modular called model generator and model solver(MGS).

MGS generates models from information provided by the users through GUI, and then solves the models built. Hence the users need not to do any model building and solving work, this will make the system usable for non-professionals in decision and computing science.

Since all technical tasks, such as model building, parameter identification and model solving, are completed in the systems, such a DSS also enables the users to focus on the aspects which they care in decision, such as controlling the parameters which are important in their decisions.

*Remark 1.* We do not intend to develop a universal model generator to deal with various decision problems, it is not likely true that such a model generator exists. However, by equipping different model generators in a DSS, it is possible to make a

DSS multi-functional. In order to focus on the main issue of this paper, we will only consider the structure shown in Figure 2, that is, one model generator in a DSS.

## 4. AN IMPLEMENTATION OF THE NEW ARCHITECTURE

In order to show that the proposed architecture is practically appropriate and implementable, this section develops a DSS under such an architecture for helping mouse-clickers in investment.

We first introduce the models the system will build for helping users in investment.

### 4.1 Models for portfolio selection

Suppose the users want to allocate their investment funds to some equities, they care about the risk and return of investment. The users are not experts in investment, they need help in deciding how much to invest on each equity.

To model such an investment problem, we define the following notations.

$n$  : number of equities,

$T$  : length of plan period,

$p_i$  : equity  $i$ 's price at the plan period end,

$p_i^0$  : equity  $i$ 's price at the plan period beginning,

$x_i$  : investment ratio on equity  $i$ ,

$x = (x_1, \dots, x_n)$  : vector of investment ratios.

Let  $X$  be a set of feasible investment ratios, which usually takes the following form:

$$X = \{x \in R^n \mid \sum_{i=1}^n x_i = 1, x_i \in [l_i, u_i], i = 1, \dots, n\}$$

where  $l_i$  and  $u_i$  is the lower bound and upper bound of the investment ratio on equity  $i$ , respectively.

Let  $\text{Return}(x)$  and  $\text{Risk}(x)$  be the return and risk of portfolio  $x$ , respectively. According to portfolio selection theories, an active investor selects a portfolio by solving the following maximization model:

$$\text{Max}_{x \in X} \text{Return}(x) : \text{Risk}(x) \leq \beta \quad (1)$$

while a passive investor selects a portfolio by solving the following minimization model:

$$\text{Min}_{x \in X} \text{Risk}(x) : \text{Return}(x) \geq \mu \quad (2)$$

where  $\mu$  and  $\beta$  are control parameters showing the investors' requirements on return and risk of investment, respectively.

Return(x) is usually expressed by the expected profit rate, that is,

$$\text{Return}(x) = E\left(\sum_{i=1}^n x_i \frac{p_i}{p_i^0}\right) - 1 = \sum_{i=1}^n x_i r_i - 1 \quad (3)$$

where  $r_i = E\left(\frac{p_i}{p_i^0}\right)$  is the expected value of the price ratio of equity i.

Suppose we can use m historical data of the equities in estimating the expected value of the price ratios during the plan period, and let  $p_i(j)$  be the  $j^{\text{th}}$  historical price of equity i, then we use the following formula to estimate  $r_i$ :

$$r_i = \frac{1}{m-T} \sum_{j=1}^{m-T} \frac{p_i(j+T)}{p_i(j)} \quad (4)$$

There are several ways to measure risk in investment, such as the variance and the value at risk, and it seems true that there is not a risk measure which is universally accepted. In this system, the users can choose one risk measure from the following three: expected loss rate, variance and VaR.

We will only explain expected loss rate as risk measure here, and refer to [3] for the variance and VaR.

Let expected loss rate of equity i be  $\text{loss}_i$ , then loss rate of portfolio x can be expressed as

$$\text{risk}(x) = \sum_{i=1}^n x_i \text{loss}_i \quad (5)$$

We use historical price data to estimate loss rate of each equity, that is,

$$\text{loss}_i = \frac{1}{k_i} \sum_{j=1}^{m-T} \frac{|\Delta p_i(j+T)|}{p_i(j)} \Big|_{\Delta p_i(j+T) < 0} \quad (6)$$

where  $k_i$  is the number of cases that a loss happened to equity i during time period T in the historical data.

When the investors use expected loss rate to express their risk preferences models (1) and (2) are linear programming models, we use some standard algorithm in solving these models.

When the users use variance in expressing their risk preferences models (1) and (2) are nonlinear programming models, we use some standard algorithm for solving these models. When the investors use VaR to express their risk preferences,

models (1) and (2) are complicated optimization models, we use the method suggested in [4] to solve model (1) and model (2). Details of these algorithms and methods for identifying necessary parameters are omitted here.

## 4.2 A DSS for portfolio selection

We implement the DSS in the platform of Microsoft Window XP using the following tools.

Programming language: Matlab.

Database: Microsoft Access.

The DSS we implement is illustrated in Figure 3, which is composed of three parts.

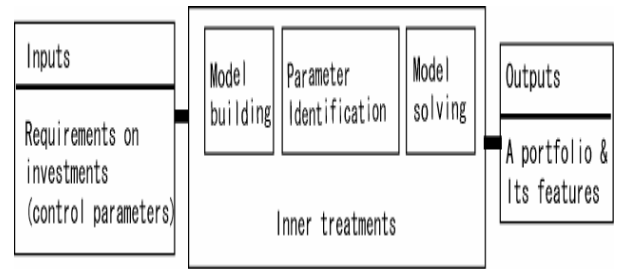


Figure 3. A DSS for portfolio selection

Part I is the GUI where the users can select all equities from a database which they like to consider in their portfolios and then input the parameters they want to control in forming a portfolio.

To facilitate the users in doing these, we divide the GUI into two windows. The users can check historical price information of all equities in the database and pick up the equities they may possibly incorporate into their portfolios in the GUI window No.1, and set the control parameters in the GUI window No.2, as shown in Figure 4.

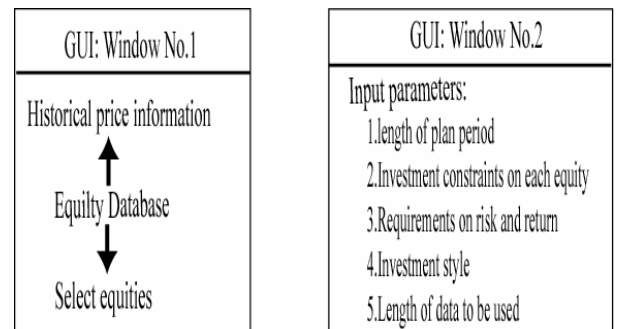


Figure 4: GUI of the DSS

Part II builds models (1) or (2) according to the investment style specified in Part II, and then identifies parameters and solves the model with a

corresponding solver.

Part III outputs the selected portfolio and risk and return of the portfolio, as shown in Figure 5.

Output window
1.Portfolio selected
2.Risk and return of the selected portfolio

Figure 5. Output of the DSS

The users can go back to part one to change their choices if they are not satisfied with the results.

### 4.3 An Experiment

We do a portfolio selection experiment using the system.

The equity database used contains the historical price data (1990~2004) of the 30 component equities of the DOW JONES INDUSTRIAL index, so our portfolio is to be formed from these 30 equities.

In the GUI window No.1, we check the historical price of each equity and the average return of the equity in some specified time period, then choose the equities to be possibly incorporated into our portfolio.

Figure 6 shows one instance, where the following seven equities were chosen as possible stocks in our portfolio:

- C(Citigroup),
- GE(General Electric),
- GM(General Motors),
- IBM,
- JNJ(Johnson and Johns),
- MSFT(Microsoft),
- JPM(JP Morgan Chase).

In the GUI window No.2, we input the following parameters for control the risk and return of investment,

- Length of plan period:  
12 months.
- lower and upper bounds of investment ratio on each equity:  
The lower bounds for C, IBM, JPM, MSFT are set to 10%, while all other bounds are the default values.
- length of data to be used in estimating the inner parameters:  
Eight years in the past.
- Minimum return:  
10%.
- Maximum risk:

- 10% in term of expected loss rate.
- Investment style:  
Passive, which means minimizing the risk while keeping the return above the specified minimum return in determining a portfolio.



Figure 6. GUI Window No.1

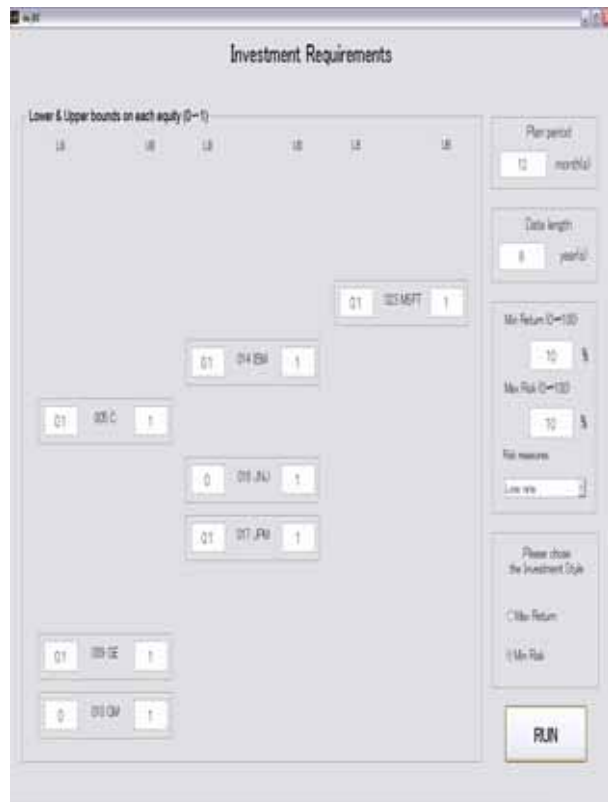


Figure 7. GUI Window No.2

When all these parameters are set, the system

generates an optimization model (2), and solves the model built with the standard solver of Matlab.

List 1 shows the model generated, which contains variables whose values are set in the GUI windows and parameters identified using (4). The objective function file generated is shown in list 2, where the coefficients are obtained from using (6).

```
% objective function of model (2)
function [rkf_opx, rkf_fv] =
rsk_opt_fun(num_x, re, myu)

% Starting Guess
x0 = zeros(num_x, 1);

% Linear Inequality Constraints
global ck_val;
selected = [];
for j = 1:30
    if ck_val(j) == 1
        selected = [selected; j];
    end
end
lin_con = [];
for i = 1:length(selected)
    lin_con = [lin_con, re(selected(i))];
end
A = [lin_con];
b = [1+myu];

% Linear Equality Constraints
global ck_val;
n = sum(ck_val);
Aeq = ones(1, n);
beq = [1];

% Lower and Upper BOUND
global lower_b;
lb_con = lower_b;
global upper_b;
ub_con = upper_b;

% Options
my_opt = optimset('Display', 'final',
'LargeScale', 'off');

% Optimization k_fun
[rkf_opx, rkf_fv, exitflag, output, lambda,
grad, hessian] = fmincon(@rsk_obj_fun,
x0, A, b, Aeq, beq, lb_con, ub_con, [], my_opt);
```

List 1: Model (2) generated

```
function f = rsk_obj_fun(x)

f=x(1)*2.255139e-001+x(2)*7.460318e-001+
x(3)*3.008323e-001+x(4)*2.668323e-001+
x(5)*2.690087e-001+x(6)*4.667637e-001+
x(7)*3.842206e-001;
```

List 2: Objective function generated

The last part in List 1 calls the solver for solving the generated model. The result is shown in Figure 8.



Figure 8. Output of the DSS

The suggested portfolio is as follows:

Equities	Investment ratios
Citi Group	0.6
General Electric	0.1
General Motors	0.0
Johnson and Johns	0.1
Microsoft	0.1
JP Morgan Chase	0.1

The estimated return of this portfolio is 12.37% and the estimated risk is 0.32%, these figures are based on the price data in the past eight years.

## 5. CONCLUSIONS

We proposed the notion of DSS for mouse-clickers, and suggested an architecture for these DSSs. Furthermore, we showed the new architecture is practically appropriate and implementable by developing such a DSS for investment.

DSSs developed under this architecture complete all technical tasks within systems for making a good decision, such as parameter identification, model building and computations, no modeling or programming work for their users.

The merits of such DSSs are as follows:

(1) Most people are able to use such systems. The users need only to be able to do clicking with a mouse and figure input from a keyboard.

(2) The users can focus on their problems instead of the technical issues by controlling parameters which are important in their decisions.

Since technical treatments are done in a black box in the users' eyes, such DSSs may need to get certain certification from some authoritative organizations, in order to make people feel comfortable with the results produced.

## Acknowledgement

This work was supported partially by the Japan Society for the Promotion of Science under grant 17500184 and partially by a research grant from Chiba Institute of Technology in 2005.

## REFERENCES

- [1] Power, D.J. (2003), A Brief History of Decision Support Systems. DSSResources.COM, [DSSResources.COM/history/dsshhistory.html](http://DSSResources.COM/history/dsshhistory.html), version 2.8, May 31, 2003.
- [2] Marek J. Druzdzel and Roger R. Flynn (2002), Decision Support Systems, included in Encyclopedia of Library and Information Science, Second Edition, Allen Kent (ed.), New York: Marcel Dekker, Inc., 2002
- [3] Dowd, K. (2002), An introduction to market risk measurement, John Wiley & Sons, 2002.
- [4] Xu, C. (2004): A Soft Approach for VaR Based Portfolio Optimization, Proc. of the Fall National Conference of Japan Society for Management Information, 112-115, 2004.