CITMOD
A Tax-Benefit Modeling System for the average citizen

By Philip Truscott, PhD

Abstract

Governments and research institutes often create computer models to play "what if" games with tax and social benefit policy. Typically, a major survey of income and expenditure, such as the Philippine Family Income and Expenditure Survey, is used. The computer models calculate the effect of policy change at the level of individual household. The thousands of households in the survey are then "multiplied up" to the millions in the national population. Usually the people able to use these models constitute a tiny technocracy of computer programmers.

This document explains the creation of a different type of computer model. The overwhelming majority of computer programming investment goes into creating information tools to ***simplify the process of tax-benefit modeling*** *rather than creating a specific tax-benefit model. It involves a* ***Policy Simulation Toolkit****. This toolkit creates a more user-friendly* ***Model Creator*** *program. The Model Creator program requires at the skill level of an intelligent layman/woman. The program's users can access a computer language, the* ***Policy Simulation Language*** *that is a midway stage between natural language and a programming language. The Model Creator then converts the tax-benefit rules into the simplest software output: the* ***Citizen Model****. This allows end users to select or de-select various taxes and benefits. The Citizen Model does not involve any programming but users can change numeric parameters such as the percentage rates of tax or weekly benefit amounts. They can also select or de-select major policy options such as entire new taxes or benefits. The fact that the Model Creator and the Citizen Model are designed for non-computer programmers represents a decentralization of power and is the justification for the title "****Democratic*** *Tax-Benefit Model."*

Table of Contents

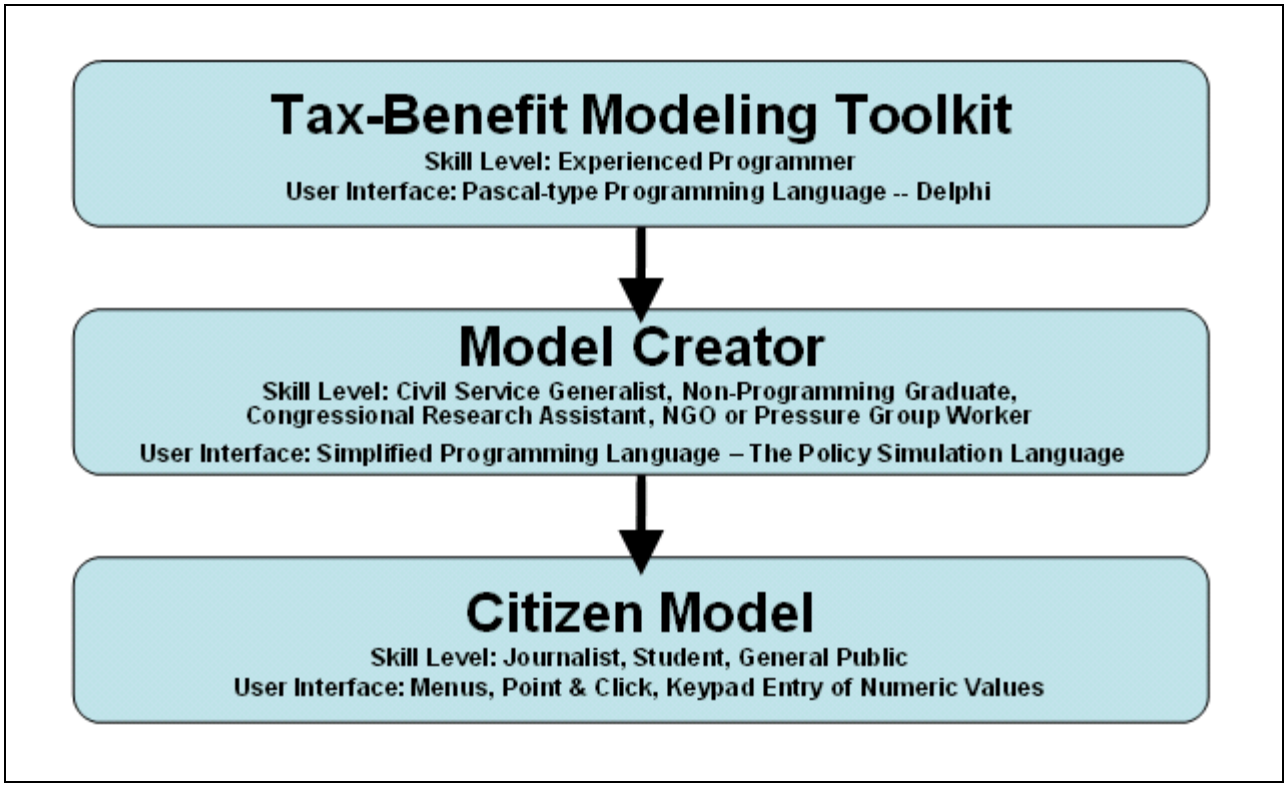
Background: A Short history of CITMOD

The current system started as a suite of programs to demonstrate the UK Tax-Benefit system in 1994. It then used Turbo Pascal programs running under the operating system MS-DOS. It represented an investment of approximately two years of work. The current project adapted Filipino survey data for use by the same system. Modernization for the Windows operating system required the most popular Windows version of the same computer language as the original model: Delphi 7. Delphi would probably not have been the language of choice if starting from scratch. However, the selection of a different computer language would be tantamount to recreating the entire system requiring about two years of work.

The original modeling system, formed part of the author's PhD thesis at the University of Surrey in the UK: "A Pluralist Model of Tax-Benefit Policy." At that time, its data source was the UK Family Expenditure Survey, which has a similar function to the Philippine Family Income and Expenditure Survey (FIES). The author wrote a review of similar systems in use by government agencies and private research bodies: "Computer Models of Tax-Benefit Policy."[1]

After the PhD, the system helped to design a Basic Income System* for the UK and predict its effects on unemployment. The plan was published by a pressure group: the Movement for Christian Democracy[2].

Figure 1: Skill Levels required for different elements of the System



* The Term "Basic Income" means a system of cash benefits paid to all adults whether in work or out of work. In developed countries such plans usually propose they would unify and replace unemployment benefits for the jobless and personal tax allowances for those in work.

Required Skill Levels

To understand how the model differs from traditional tax-benefit models Figure 1 illustrates the different levels of skill required by different parts of the system.

The Modeling Toolkit

The second phase of the research in 2008-09 involved converting the existing Pascal code to the most modern form of the same programming language, Delphi, to give the program a Windows appearance. Some parts of the existing DOS program were replaced by royalty free software components so produce Excel Spreadsheet tables.

The Modeling toolkit is a set of programs, which, when compiled, produce the Model Creator program described below.

Figure 2: Programming Taxable Income in Pascal and the Policy Simulation Language

```
      PASCAL:
1
2     If (Total_Income > Taxable_Allowances) then
3         Taxable_Income := Total_Income - Tax_Allowances;
4     Else
5         Taxable_Income := 0;
6
7     PSL:
8
9     Taxable_Income = Total_Income LESS Tax_Allowances.
```

The Model Creator Program

The Model Creator program allows the user to specify new taxes and benefits using a computer programming language: The Policy Simulation Language (PSL). As will be shown below this has been designed to make programming tax-benefit rules as close as possible to natural language.

Tax-Benefit Modeling Operators

To illustrate this Figure 2 compares statements for programming the concept of "Taxable Income" using PSL and C++. Many programmers would require four lines of computer programming to specify the calculation for taxable income (see lines 2-5 of Figure 2). PSL achieves the same thing through a special mathematical operator "***LESS***". This operator is simply a subtraction operator whose result may not be less than zero. It would of course be possible to achieve a similar result in a traditional programming language by using a more condensed programming style but this would make the programming statements even harder for a nonprofessional to understand. By contrast, the PSL statement is close enough to plain English that it could be understood by an accountant or payroll worker.

Wherever possible, PSL uses special operators to reduce the programming requirements and to produce programming statements that are clear enough to be read and checked by non-programmers. Often social welfare benefits are tapered as income rises. In the 1970s, UK government introduced a Family Income Supplement that provided a payment that was reduced by 50% of the family's weekly earnings. Such a definition might appear like this in PSL:

```
Family_Income_Supplement =
            ($10 * Number_of_Children) WITHDRAWN_BY 50% of Total_Earnings.
```

As with the "LESS" operator the "WITHDRAWN_BY" operator cannot produce a result less than zero. Without this feature, it is very easy for a careless programmer to specify a welfare benefit that accidentally turns into a "tax" because the negative values have not been allowed for.

Figure 3: Programming Child Tax Allowances Using C++ and the Policy Simulation Language

```
     C++:

 1   Child Allowances = 0;
 2   for (int counter=1; counter<=People_In_Family; counter++)
 3   {
 4        if ((Relation_to_Family_Head = Child) and (age <= 18))
 5            Child_Allowances = Child_Allowances + 8000;
 6   }
 7
 8   PSL:
 9
10   Child_Allowances = 8000 *
11       NUMBER_IN_FAMILY (Relation_to_Family_Head = Child and age <= 18).
```

The Avoidance of Programming Loop Structures

One of the most common programming errors is the "infinite loop." PSL avoids this and other iteration errors by providing functions that obviate the need to write looping constructs. Figure 3 illustrates how PSL avoids the use of looping structures with a function that operates on an entire household. Lines 2-6 of the example count the number of children using a traditional for loop. Line 11 illustrates the use of a function that counts the number of people in a family for whom a logical condition is true.

Figure 4: Household and Family Functions in the Policy Simulation Language

NUMBER_IN_HOUSEHOLD	This counts the number of people in a household for whom a given logical condition is true.
NUMBER_IN_FAMILY	This counts the number of people in a family unit for whom a given logical condition is true.
ANY_IN_HOUSEHOLD	This returns a TRUE value if any person in the household matches the logical condition in brackets. Otherwise, it returns false.
ANY_IN_FAMILY	This returns a TRUE value if any person in the household matches the logical condition in brackets. Otherwise, it returns false.
TOTAL_HOUSEHOLD	The sums the value of a mathematical expression across the household.
TOTAL_FAMILY	The sums the value of a mathematical expression across the family.

Figure 4 shows the six household and family functions used for writing tax-benefit simulations. For the purposes of tax-benefit calculations a "household" consists of all people in the same dwelling. The term "family" corresponds to the term "Benefit Unit" or "Tax Unit" in the UK. It refers to an adult, his or her spouse, and any dependent children of that adult. It corresponds to the family unit definition used for Filipino Income Tax (such family units may also include a dependent senior citizen adult). Thus there may be several family units within one household.

Family and Household Relationship Pointers

It is quite common for tax and benefit rules to refer to the characteristics of particular family or household members. For example, the rules for the UK's Housing Benefit pay a more generous payment if the Head of Household is disabled. To find the head of household a computer program might require a complex "for" loop or a complex condition. The Policy Simulation Language avoids the need for this with a built in function, which identifies the Household Head with a single keyword: "Head_of_Household's." This enables the following type of statement to be used in PSL:

```
if (Head_Of_Household's ability_status=Severely_Disabled)
```

The full list of household and family relationship pointers is:

```
Husband's
Wife's
Head_of_Tax_Unit's
Head_of_Household's
```

The apostrophe + "s" syntax serves to make the PSL language statement much closer to plain English than programming statements usually used for this purpose.

Automatic Menu-Creation

The Model Creator program has been designed to produce an easily understandable set of menus that can be accessed from the Citizen Model end user program. To illustrate how this works consider the definition of Income Tax shown in Figure 5. In 1999 Filipino Income Tax had seven rates ranging from 5% to 34%[†].

Figure 5: A Definition of Income Tax in the Policy Simulation Language

```
 1 {START_TAX Filipino Income Tax}
 2
 3 1st_Level         = PhP   10000
 4 2nd_Level         = PhP   30000
 5 3rd_Level         = PhP   70000
 6 4th_Level         = PhP  140000
 7 5th_Level         = PhP  250000
 8 6th_Level         = PhP  500000
10
11 1st_Tax_Rate          =  5 %
12 2nd_Tax_Rate          = 10 %
13 3rd_Tax_Rate          = 15 %
14 4th_Tax_Rate          = 20 %
15 5th_Tax_Rate          = 25 %
16 6th_Tax_Rate          = 30 %
17 7th_Tax_Rate          = 34 %
18
19 Current_Income_Tax =
20   (Taxable_Income LESS    6th_Level               * 7th_Tax_Rate) +
21   (Taxable_Income BETWEEN 5th_Level AND 6th_Level * 6th_Tax_Rate) +
22   (Taxable_Income BETWEEN 4th_Level AND 5th_Level * 5th_Tax_Rate) +
23   (Taxable_Income BETWEEN 3rd_Level AND 4th_Level * 4th_Tax_Rate) +
24   (Taxable_Income BETWEEN 2nd_Level AND 3rd_Level * 3rd_Tax_Rate) +
25   (Taxable_Income BETWEEN 1st_Level AND 2nd_Level * 2nd_Tax_Rate) +
26   (Taxable_Income BETWEEN 0         AND 1st_Level * 1st_Tax_Rate).
27 {END_TAX}
```

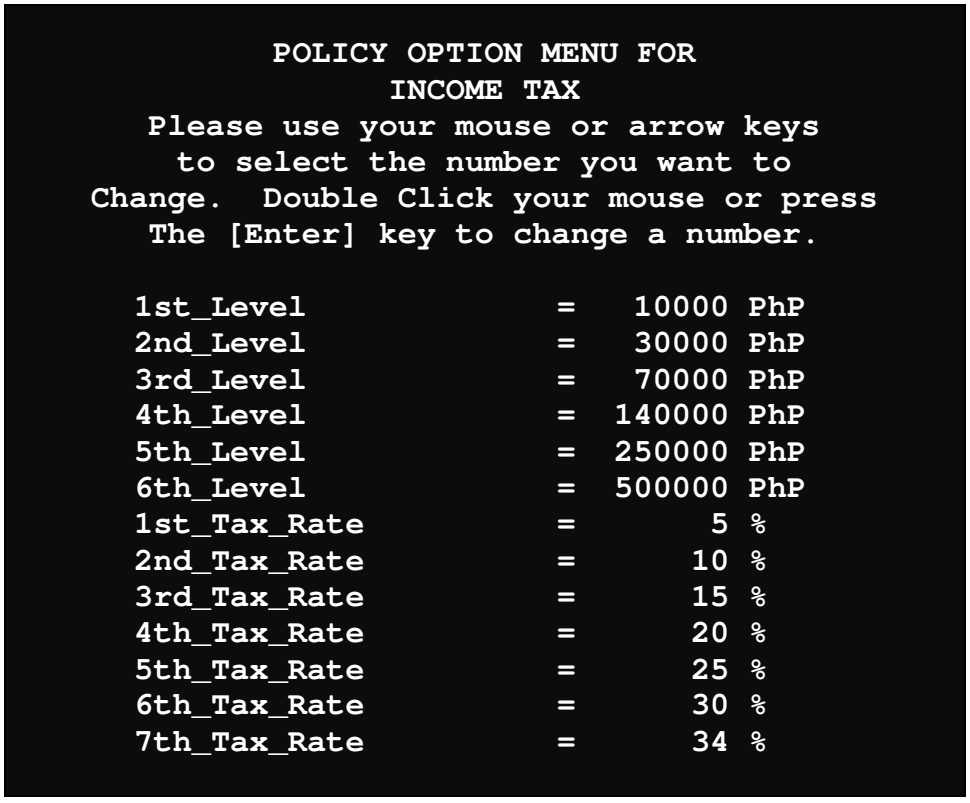
[†] According to BIR Form 1700 October 2001 (rates were scheduled to change).

Often the tax rates are "hard-coded", meaning that the actual number 0.05 would be inserted directly into the original computer program for the starting rate of tax.

When the Model Creator program compiles the statements in Figure 5 it creates not only the programming language statements to calculate Income Tax for each family unit. It also builds a user interface within the Citizen Model to allow non-programmers to play "what-if" games with the numbers. Specifically it:

- Creates a menu of tax rates and tax levels
- It preserves the % symbol or currency symbol to show the values in the most user-friendly way (in the Citizen Model the starting tax rate is shown as 5% rather than the number 0.05)

Figure 6: Citizen Model Computer Screen for Changing Income Tax Parameters (DOS Version)



When the Model Creator program compiles a new model all the labeled constants in Figure 5 on lines 3 to 18 become menu options in the end-user program.

All the numeric options are grouped together on the same screen because they are written between the programming keywords "START_TAX" and "END_TAX."

Similar keywords are used to group parameters relating to social benefits: "START_BENEFIT" and "END_BENEFIT."

Natural Language Constant Recognition

Typically programmers use sample surveys and embed the category numbers into programming statements. For example, suppose the number "2" means "dependant child" in a variable that records a person's relationship to the head of household. Programmers frequently put the number "2" directly into the code making the program difficult to read. Accountants, academics or policy researchers are forced to refer to a codebook to check such statements. PSL, however, requires that a full category label be placed into the tax-benefit declaration. Consider the definition below:

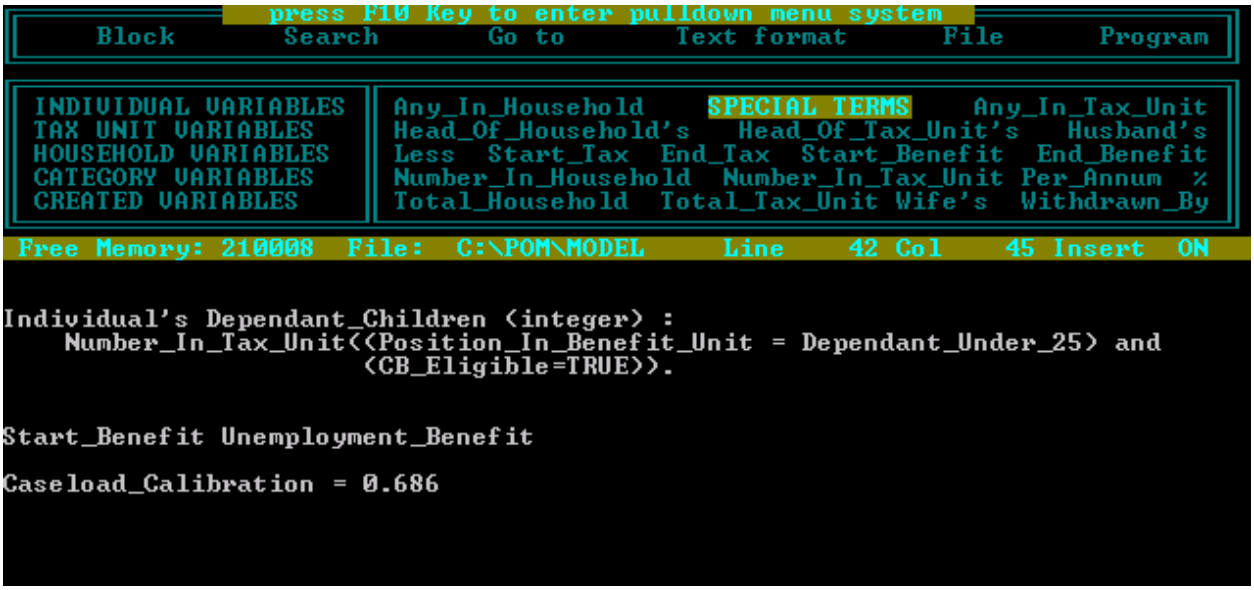
```
Individual's Dependant_Children (integer) :
    Number_In_Tax_Unit((Position_In_Benefit_Unit = Dependant_Under_25) and
                       (Child_Benefit_Eligible=TRUE)).
```

The term "Position_In_Benefit_Unit = Dependant_Under_25" is much more meaningful (and less error prone) than "Position_In_Benefit_Unit = 3". This process enforces a "self-documenting" style of programming that makes the tax-benefit rules more susceptible to peer review. Changes of staffing need not be fatal to the model since the rules are inherently readable.

Model Creator User Interface

The main screen of the Model Creator program is shown in Figure 7 below.

Figure 7: User Interface of Model Creator Program (DOS Version)



There are four main areas of the User Interface as follows:

- The Pulldown Menu Bar (Block, Search, Go to, etc.) contains standard text editor functions. The user can search for words, copy and past blocks of text.
- The Special Terms Menu (includes the special PSL language terms like "Total_Household" etc.) . If the user double clicks on one of these special keywords then it will be inserted in the text editor window at the current cursor position.
- The Variable Picker (These are the five pop-up lists grouped under the word "INDIVIDUAL VARIABLES"). The user can click on one of these lists to display lists of variables. If the user

double clicks on a variable name in a pop-up window then it appears at the current cursor position in the editor window at the bottom.

- The Text Editing Window (The area under the orange line that starts with "Free Memory" in Figure 7). This is where the users build up their definitions of taxes and benefits.

Compiling the Tax-Benefit Rules

Once the rules have been written, users can "parse" the rules using the "Program" sub-menu. Syntax errors will be revealed with pop-up error messages. If the model is correctly specified then a new version of the Citizen model will be generated.

Citizen Model User Interface

Figure 8: Main Menu Options of 1994 Version of Citizen Model

If users choose either of the first two options they can select or deselect an entire tax with a minimal number of keystrokes or mouse clicks. For example if the user selects the benefit definitions then the following menu will be displayed:

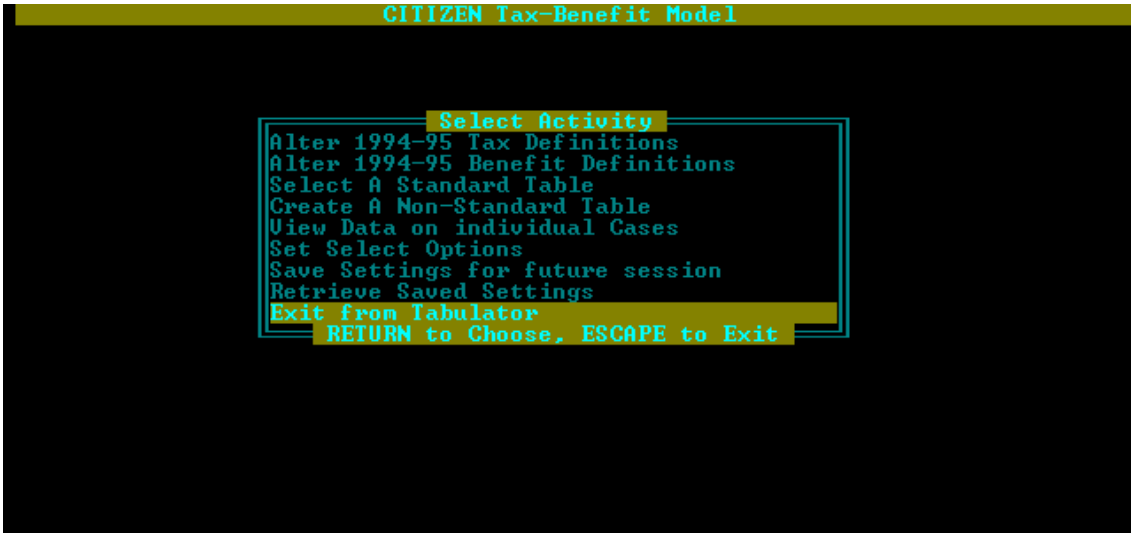
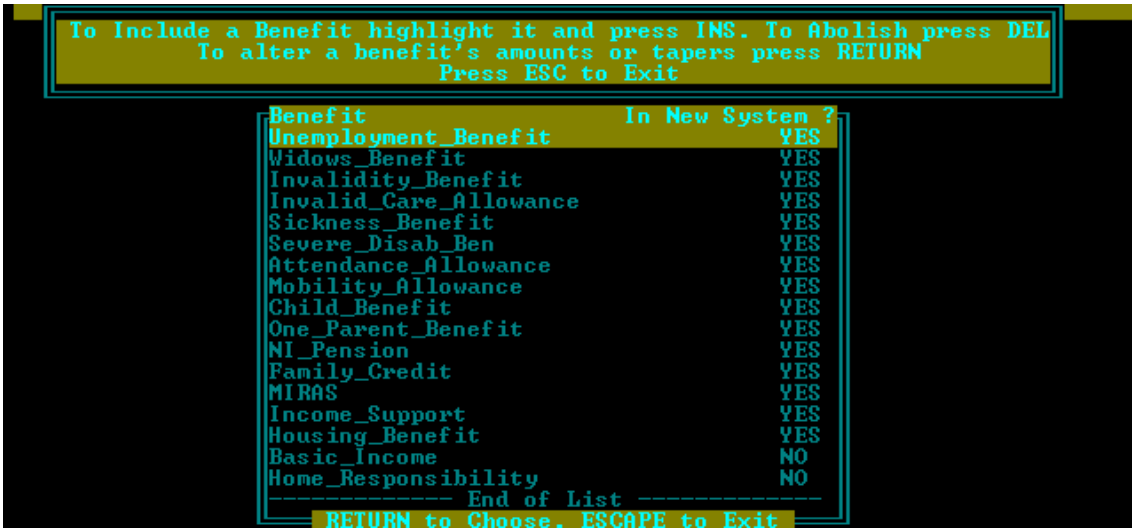
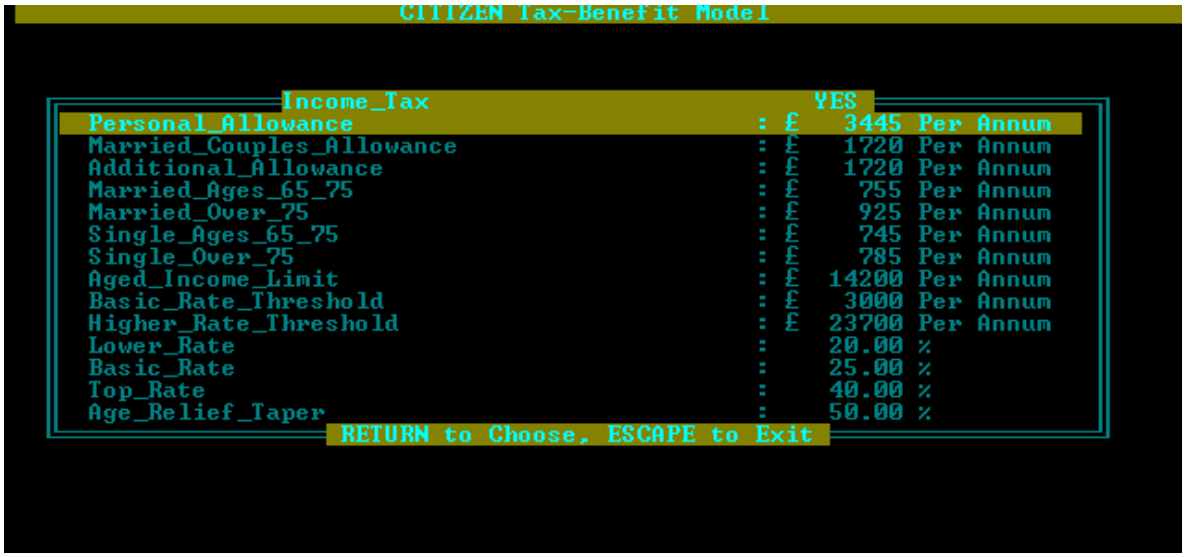


Figure 9: Master List of Social Benefits



Using the menu shown in, model users can select or de-select an entire tax or benefit with the "INS" or "DEL" keys. If the user selects a specific tax or benefit by pressing the "Enter" key then all the related percentages and monetary values may be altered. Figure 10 shows a menu that has been generated automatically from the PSL statements for UK Income Tax. The first line of the menu shows the "Personal Tax Allowance". This is the income below which no individual pays Income Tax; it is equivalent to the RP 8,000 Peso tax threshold. The user can increase or decrease any of these values by pressing the "Enter" key.

Figure 10: Citizen Model sub-menu allowing changes to Income Tax

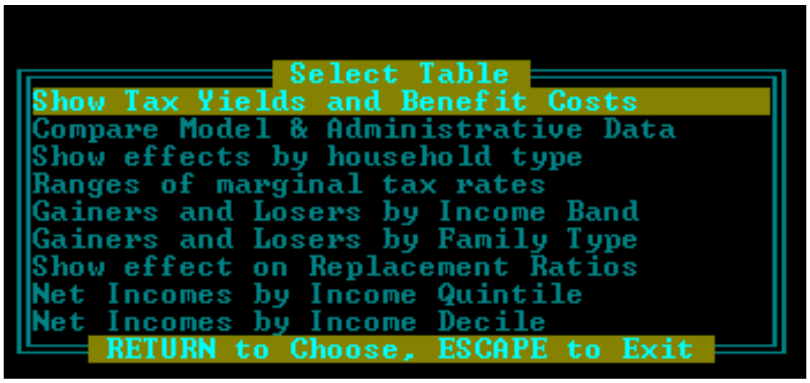


Outputs and Analysis Tables

The Citizen Model's Standard output tables are shown in Figure 11 below. All of the options shown in Figure 11 should be self-explanatory except the following:

- "Compare Model and Administrative Data" shows how closely total tax yields and benefit costs for the national population compare to government totals. The government totals are shown side by side with the Citizen Model's estimated totals.
- "Ranges of Marginal Tax Rates" shows the percentage rate of tax on earning an extra Peso of income under the current tax-benefit system and the system as altered by the current model user.

Figure 11: Standard Output Tables from the Citizen Model



If the user selects any of these tables then the Citizen Model will calculate the effect of any tax or benefit changes entered by the user. The model will compute the net incomes, taxes and benefit for each household in the database and multiply up the results to the national population.

Figure 12: National Tax Revenues and Benefit Costs of a 10% Rise in Income Tax

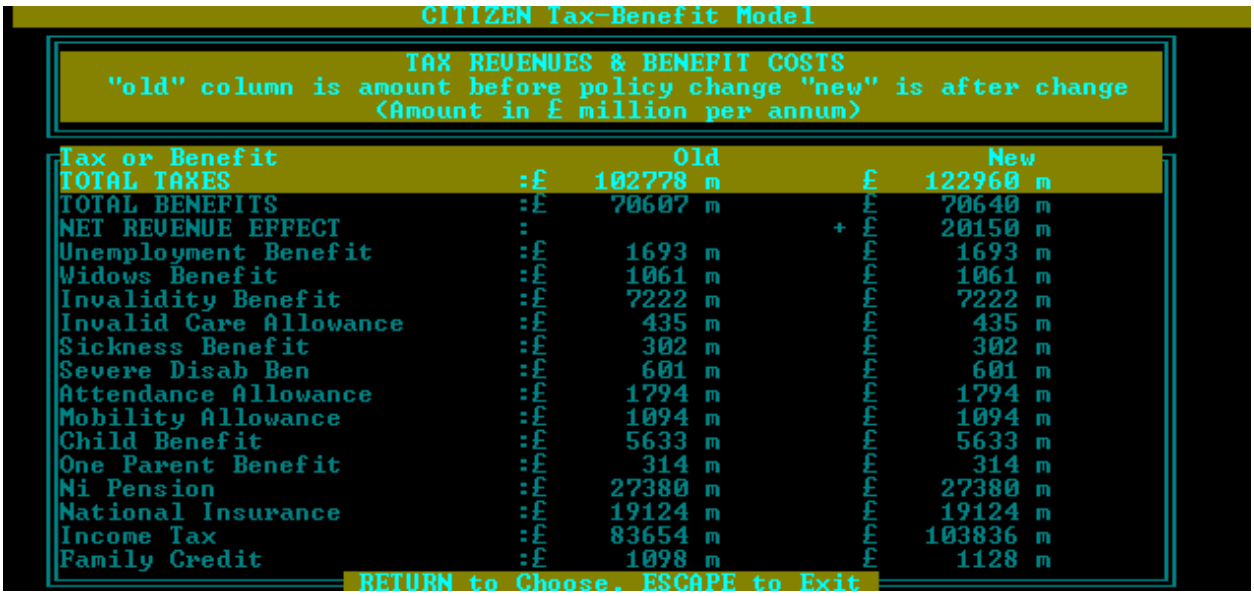


Figure 12 above shows the effect of increasing the main UK rate of Income tax from 25% to 35%. The top line of the table shows the total tax revenue before and after the policy change. The third line shows the net revenue effect of the policy change (in this case a revenue gain of 20,150 million).

Saving Policy Settings

During a session using the Citizen Model it is possible that the user might change literally dozens of policy parameters (tax rates, benefit amounts etc.). On exiting the Model the program prompts the users to save their settings. The saved settings can then be read into to the Model during a future session.

Data Preparation – Recovery of Tax Units

If you are about to use the modeling environment you must perform data preparation first to group the individuals into family units. For the purposes of the CITMOD for the Philippines in 2008-09 the following steps were required. Data Preparation

The data used was the most recent year of data of the Philippine Family Income and Expenditure Survey (FIES) 2006. The FIES is a follow on survey that asks more detailed income and expenditure questions to people who respond to the Labour Force Survey (LFS). The database for the 2006 FIES and the matched individuals from those households was purchased from the National Statistics Office (NSO). The household interviews for the FIES took place throughout 2006, with a final round of interviews in January 2007. Since the data collection fell almost entirely within 2006, it was decided to replicate the Philippine tax system in 2006 for the purposes of this research. The total matched sample contained 198,670 individuals. However the number of individuals where there is matched household income data is somewhat less than this. The people in households with full data numbered 189,079. These people formed the base population for this research. Using the NSO's standard procedure to multiply up the sample cases to the national population, these people represented 85.4 million Filipinos in 12.8 million households. This was slightly less than the NSO estimate of the Philippine population in 2006, so an alternative grossing up scheme was used.

Recovering Tax Units

The FIES suffers from a defect that complicates the process of tax-benefit modelling. The FIES survey questionnaire records the relationship of each person in a given household to the head of the household but not to the head of the family (as families are defined for the purposes of income tax). By contrast the UK's Family Expenditure Survey (FES) the survey distinguishes between the head of the family and the head of the household. The facilitates the work of tax-benefit modelling because there is clear evidence within the survey about which Income Tax payers are married and which children within the household are theirs for the purpose of child tax allowances.

The same problem was addressed when a tax-benefit model was constructed for South Africa: SAMOD. Wilkinson (2009) describes the process of placing children into family units as follows:

> "... a likely carer was identified for each child under the age of 18. This method assigns children to carers according to a series of rules. First, children are assigned to the first woman in the household who is aged between 13 and 40 at the child's birth. If no carer is assigned then the child is assigned to the first woman aged over 40 at the child's birth. If the child still has no assigned carer then the process is repeated for men aged 13 to 40, then, men aged over 40."

CITMOD followed a similar process but tried to fine tune the allocation by finding the household members who were the at the most plausible ages in relation to each other. An analysis of the data revealed that a significant number of Filipinos live in single family households. These are defined as households where individuals only have the following values for the variable that describes the relationship to the head of household:

a) Head
b) Wife/ Spouse
c) Son / Daughter

An analysis of these households allowed typical age gaps to be calculated as follows:

a) Average age gap between a mother and the children living with her : 28 years;
b) Average age gap between a father and the children living with him: 31 years.

If a person's marital status variable showed that he or she was married then a partner was allocated from the household who was:

a) Also recorded with a married marital status;
b) Not previously allocated by the program as the spouse of another person;
c) The available person who was closest in age to the person being matched.

In the statements below the term "dependent child" is the definition used for Filipino income tax.

After the matching of the spouses the children in the data set were allocated to family units according to the following steps (which were taken in order):

a) Children with a relationship to head of household of "son/daughter" were allocated to the head's family unit;
b) Children with a relationship to head of household of "Grandchildren" and where there were suitable children of the head in the household were allocated to those adult children;
c) Children with a relationship to head of household of "Other Relative" and where there were suitable adult "Other Relatives" in the household were allocated to those adults;
d) Children with a relationship to head of household of "Non Relative" and where there were suitable adult "Non Relatives" in the house were allocated to those adults;
e) Where the household had adult sons of the head of household he was matched with the woman closest in age who had the relationship to head of household status of "son/daughter in law" - any unallocated children in the household were matched with this mother;
f) Where there were possible mothers who were the daughters of the head of household (possible single mothers) any unallocated children were allocated to the potential mother closest to the average mother-child age gap;
g) Where there were possible fathers who were the sons of the head of household (possible single fathers) any unallocated children were allocated to the potential fathers closest to the average father-child age gap;
h) Where there were possible fathers who were the other relatives of the head of household (possible single fathers) any unallocated children were allocated to the potential fathers closest to the average father-child age gap;
i) Where there were adult siblings of the head of household any unallocated children were matched with these persons if their status was "other relative" based on the age gap method;
j) Where there were fathers or mothers of the household head they were matched into family units where there was another opposite sex household member with the same status;
k) Where there were boarders of the head of household any unallocated children were matched with these persons if their status was "non relative" based on the age gap method;
l) Where there were domestic helpers of the head of household any unallocated children were matched with these persons if their status was "non relative" based on the age gap method.

This method produced 71, 645 family units in the data file.

Testing the Model: Micro-validation

Micro-Validation – a 'Double Programming Approach'

In order to check the accuracy of the CITMOD tax-benefit calculations a rigorous cross-checking approach was used at the level of individual tax-payers in the survey data as follows:

a) A data file was prepared from the matched FIES-LFS data after the family units and individual earnings data had been derived by the process described above;
b) For each person in the data set there was a record of the individual's earnings, number of children, and tax-payer status (single, married or head of family);
c) Income Tax, SSS contributions and PhilHealth contributions were calculated using CITMOD's own native language: the Policy Simulation Language (which is a derivative of Pascal);
d) Income Tax, SSS contributions and PhilHealth contributions were calculated by a different person using a program written in the Structured Query Language (SQL) in Microsoft Access;
e) The three sets of tax bills were compared on a case-by-case basis to insure the same values were produced by the different programs;
f) Where different tax bills were discovered discussion between the two programmers led to programming revisions until the two programs produced matching tax bills.

Art Tan of the Ateneo School of Management had previously maintained payroll software for a small company of 300 people. He imported the subset of the FIES-LFS data into Microsoft Access and wrote procedures to calculate Income Tax, PhilHealth and SSS contributions.

Figure 13: Example PhilHealth Lookup Table

SalaryUpto	SalaryBase	EmployeeShare
4,999.00	4,000.00	50.00
5,999.00	5,000.00	62.50
6,999.00	6,000.00	75.00

Micro-Validation: PhilHealth

At the end of the comparison process very small differences remained between the CITMOD and SQL tax bills for PhilHealth payers. This is the result of the fact that CITMOD uses a percentage method for calculating the tax. Tax bills are calculated by multiplying the PhilHealth taxable income by the tax rate of 1.25%. In the case of the SQL calculations a lookup table method is used. This process is illustrated by **Error! Reference source not found.** above. Consider a worker who falls within the monthly salary range 4,000-4,999 pesos. He or she will have a deduction of exactly 50 pesos. When his or her salary exceeds 5,000 pesos the next standard contribution of 62.50 is deducted. There are 21 PhilHealth lookup income groups until the salary reaches 25,000 pesos when the maximum contribution is deducted.

In theory, CITMOD could have been re-programmed so that it also used the lookup table method but this would have made it a less user-friendly program. Currently a policy-maker only needs to change one tax rate to simulate a new policy. The difference between the percentage method and the lookup method was not negligible in the case of PhilHealth contributions. In the case of the PhilHealth lookup table, the standard deductions tend to skew the tax bills downwards compared to the percentage method. For each tax bracket the tax bill is not always the tax that would be paid by a person with an income in the middle of the salary range, but a person at the lower bound of that range. Across all persons in the matched dataset this produced average CITMOD PhilHealth bills 5.15% higher compared to the SQL calculations.

Micro-Validation: Social Security

Payroll managers also often use a lookup table for Social Security Contributions. In this case the differences with the percentage method were trivial. The lookup table tax values are calculated for earners in the middle of each salary range. The total SSS bills were thus almost identical for the CITMOD and SSS methods. The total values were within one tenth of 1% of each other (0.059%).

Micro-Validation: Income Tax

In the case of Income Tax the lookup table method is not used, but the small differences in PhilHealth and SSS contribution calculations in turn produced small differences in the SQL and CITMOD values for Income Tax. Because these deductions reduce taxable income they also produced small differences in tax bills. However these effects were small in proportion to total taxable income. Because CITMOD's PhilHealth contributions were slightly higher, its Income Tax bills were slightly lower by about one fifth of 1 per cent (0.225%).

Some Built-In Policies: A Software Policy Almanac

The current version of CITMOD includes built-in reform options to the Philippine tax system of 2006 as follows:

Carbon Tax

Carbon Taxes are frequently advocated by environmentalist parties as a way to discourage the use of the mostly highly polluting fuels. It would be comparatively easy to model a Carbon Tax using the Philippine Family Income and Expenditure Survey, which includes spending on Gas, Diesel, LPG, charcoal. The model building should exercise car to make a Carbon Tax add to the transport costs of non-car owners (bus and jeepney riders). Since Carbon Taxes might be extremely regressive they are sometimes linked with Citizen Income schemes (see below) to protect the poor from their price effects.

Basic Income

A Basic Income[*] would give each adult citizen a cash payment as of right. In developed countries it is often proposed as a solution to the "why work" problem posed by Unemployment Benefits. According to this argument workers who received welfare payments while unemployed will have no incentive to accept a many low paid jobs. After losing their welfare benefits and paying income tax they are little or no better off than if they stay on welfare. With a Citizen's Income (CI), they would keep the cash payment but they would be taxed on the first Peso of income (since the CI would replace personal tax allowances).

In developing countries the "why work" argument for CI is weaker because welfare benefits for the unemployed are rare or non-existent. Some have argued that work incentives would be weakened as the unemployed could simply subsist on CI and pass up lower paid jobs. Others have argued that any disincentive effects would be outweighed by the positive stimulus to impoverished rural economies. On a much smaller scale it would have the demand-boosting effects of Roosevelt's New Deal. Two developing countries with significant public debates on CI include South Africa and Brazil, which has implemented a cash grant to about 25% of its poorest families (the bolsa familia). Some see this as the first stage on the road to Citizen's Income, and in 2004 President Lula signed a law that will eventually establish one. However, with uncertainty over CI funding, this may represent a declaration of intent[3].

The philosophical arguments for Citizens Income center on the dignity of the individual. People will be protected from the direst forms of want, without the need to subject themselves to the indignity of a

[*] Sometimes called Basic Income, Basic Income Guarantee or Tax Credits.

means test. Archbishop Desmond Tutu has joined calls for a Basic Income by highlighting the problem of means-tested benefits:

> "The conditions often attached to social transfers tend to prevent the poorest families, the very people who most desperately need income support, from accessing grants. Universal grants ... offer one of the simplest and most effective strategies. ... Basic Income movements ... are designed to enhance the dignity, wellbeing and inclusion of all people, and to move us closer to our vision of social equity. ... Hasten the day when all will have freedom from hunger and extreme want.[4]"

Earned Income Tax Credit / Worker Credit

An Earned Income Tax Credit could be seen as a Citizen's Income for those in work. A fixed cash grant would be added to the wage payment in place of tax allowances. According to traditional arguments about incentives the effect on employment would be entirely positive since there would be an increase in the incomes of the working poor with no similar boost for the incomes of the unemployed. The USA has a payment with the title "Earned Income Tax Credit" but this is means tested (it tapers away as income rises) and has such a complex formula that it is rarely claimed or understood. The general understanding in tax-benefit policy literature is that a personal tax credit is a fixed cash payment added to wages in place of a tax allowance. The exclusion of the unemployed would make possible a much more generous payment per person.

A Flat Income Tax

Economic liberals frequently argue the case for a single rate of income tax. It would neither be regressive (taking a higher percentage of low incomes) nor progressive (taking a higher percentage from the rich); it would be proportionate. The Citizen Model could calculate the rate at which Filipino Income Tax could be revenue neutral and proportionate. If a "flat" income tax were combined with a Citizen's Income this would remove some of the moral arguments against a single tax rate. Even though the marginal rate of tax was the same for all tax-payers, the "average rate of tax" (the combined effect of subsidies and taxes) would be much more favorable for the low paid. In fact for wage earners at the bottom of the income distribution would receive more in Citizen's Income than they would pay in tax. It is argued that a flat income tax with a Citizen's Income is the only way to combine social justice and simplicity.

Unified Income Tax

This reform implies the unification of Income Tax and social security contributions on individuals to lower the administration costs of small businesses in administering tax policy. Frequently such plans are posited as an adjunct to Citizen's Income plans.

Unit Costs- A General Purpose Language extension

The Policy Simulation Language was extended to allow the inclusion of non-tax-benefit policies. For example, if the model builder includes the unit cost of a non-benefit expenditure item this could then appear as a sub-menu in the Citizen Model. Some candidates for this treatment might be:

- Varying the average cost per public school student
- Quantifying the cost per kilometer of track of adding to the Metro Manila MRT / LRT
- Purchasing a hectare of land under land reform
- Building one unit of low cost housing

This would take the Citizen Model in the direction of a more general purpose public policy model. Since the user would be able to balance the cost of these new policies in the tax side, it would lead to a more

useful form of cost-benefit analysis. A new key word was added to the Policy Simulation Language called "Unit_Cost" with the following parameters:

- A Title
- A Unit Name (such as "hectare of land")
- A Unit Cost (a value in money)
- A reference note (This would be display in the Citizen Model to indicate a book or academic paper that used for the source of the unit cost).

The effect of adding unit costs in the Model Creator program would be that the Citizen Model would then display it as a sub-menu under "Benefits." Citizen Model users could then experiment with different values for the units and unit costs. They could also balance increased costs by simulating increased taxes during the same Model run.

The unit cost option may be used either for revenue or expenditure items. If a revenue item cannot be allocated to specific persons (such as an increase in Corporation Tax) then this would be found under the menu option "**Alter Non-Income Based Taxes**". A similar policy on the expenditure side that might not involve a cash transfer to an individual (such as 20 kilometers of additional railway line) would be found under the menu option "**Alter Non-Income Based Policies**".

Unit Costs and Good Governance

Promoting wider understanding of government unit costs would also help anti-corruption journalists. The Malaysian anti-corruption campaigner, Alatas[5], commented that reporters were some of his biggest allies. The policy almanac could provide them with a convenient reference to critique government spending costs in their areas.

Unit Costs Presented in the Policy Almanac

The discussion of unit costs in Options for the Philippines would be the main context for the book to mention non tax-benefit policies. For example, a section on poverty housing could describe the cost of building one home along the differing models of Habitat for Humanity, *Gawad Kalinga* and *Techo Para Chile*[‡]. In each a case a photo of each type of home would be displayed together with the cost together with a quote from someone housed by each organization. Journalistically the quotes and photos would do much to prevent the text from becoming to dry. Most of the other service areas could be illustrated in the same way.

Complex Unit Cost Calculations

The British water supply expert Keith Stallard[6] commented on this proposal and observed that some service improvements cannot be simulated with a simple unit cost:

> "To give a few hundred new urban homes a water supply might only cost $100 US per home. All you have to do is connect them to existing pipes. If you connect thousands of homes sometimes you need to lay new major pipes. To connect one hundred thousand homes you need to start a new La Mesa Dam project."

He suggested a form of lookup table to simulate the costs of extending water supplies.

[‡] "Techo Para Chile" means "A Roof for Chile." A NGO worker from Chile explained its approach thus: "If Habitat for Humanity built cars they would be big SUVs. *Techo* build the housing equivalent of a motorbike. They can build something in a day." See: www.untechoparachile.cl

Figure 14: Illustrative Lookup Table to Calculate Water Supply Costs

Cost of Providing New Water Supplies per Home	
Number of New Homes	**Cost**
Up to 10,000 new homes	**$100**
10,000 to 50,000 new homes	**$300**
100,000 to 300,00 new homes	**$350**
Over 300,000 new homes	**$400**

The type of table shown in Figure 14 above is simple enough to include in the text of Options for the Philippines, and would be easy to program. This approach is preferred to using polynomial equations or other forms of calculation that would preclude the understanding of the general reader.

Conclusion – Public Policy Debate and Nation Building

At the time of the 2007 Philippine General Election the voters were asked to choose between two groups ("Team Unity" and the "Genuine Opposition"). Neither group appeared to have any understandable ideology. Since then, the public policy debate has been negative and reactive. After the ZTE Broadband scandal the main demand was the ouster of the president, rather than any specific reform to make government more transparent or democratic. Overseas there are examples of constructive reform movements. Single party domination was ended in the city of Sydney, through transferable voting. Primary elections weakened the power of party bosses in US elections. Publicly financed television broadcasts in the UK help equalize the power of parties at election time.

In 2007, the public debate over the Sumilao farmers seemed entirely devoid of plans to improve on the Philippines existing land reform program. This is in stark contrast this with the South African campaign for a Basic Income Grant. A broad coalition of politicians and civil society groups managed to join forces behind a specific constructive policy change. The Confederation of South African Trade Unions has joined forces with the South African Catholic Bishops Conference, other religious groups and the most important confederation of NGOs. They have influenced the main opposition party, the Democratic Alliance, to support Basic Income. The various groups managed to organize a public march with 3,000 demonstrators[7].

Policy initiative pamphlets seem to be a category of literature that is entirely missing in the Philippines. In the UK, for example, one can attend annual conferences of the main political parties. Thousands of people (mainly party volunteers) hold televised debates on policy. If the same pattern of involvement existed in the Philippines one would be able to visit such a conference and find by pamphlets with titles like "The Young Christian Democrats' plan to end poverty housing," or a Young Nacionalista paper called "Ending Unemployment through Free Enterprise."

Similar positive thinking needs to emerge in the Philippines to lift the debate from the level of popularity contests and insult trading. Support for democracy is still strong, but support for political parties is weak. According to the World Values Survey 83% of Filipinos supported the concept of democracy. However only 4% of Filipinos belonged to a political party compared to 9% in India, 12% in South Africa and 20% in the USA. When asked "which party respondents would vote for" no party was supported by 15% of the respondents[8]. Disillusion with politics is particularly striking among the young. A survey of 15 to 30 year olds showed that only 1% were members of parties, labor unions or professional organizations[9].

As Filipinos commemorate the 25th anniversary of Ninoy Aquino's death they are constantly reminded of his declaration that "The Filipino is worth dying for." But not all are convinced "The Filipino is worth voting for."

Compared to their North American and European counterparts, Filipino students lack any enthusiasm for political programs and parties. Their attitudes seem closer to the students of Vietnam where there are no multi-party elections[10]. A sense of powerlessness exerts the force of an invisible dictator.

In this climate there can few higher nation-building goals than broadcasting ideas for change. In themselves ideas cannot produce a more constructive issue-based political culture, but they can keep hope alive.

Software Installation

Run the file: **CITMOD_Installer.exe.** The installer only gives a choice of the installation disk drive, not the directory; a root directory named \POM will always be installed on the target computer.

Assuming that the c: drive is selected the installation process will create a series of directories with purposes indicated in table below.

Table 1 Directories Created by CITMOD_Installer

C:\POM	The root install directory
C:\POM\DATA	The household expenditure data and data dictionary files.
C:\POM\DOCUMENTS	This help file and an example file called FILIPINO2006.PSL which specifies some Filipino taxes in the year 2006 plus some definitions of new policy options.
C:\POM\MODBUILD	This directory contains the application file MODBUILD.EXE (the Model Builder Application) which can be used to generate end user models (in conjunction with some 3rd party software such as Delphi 7 and two components used for mapping and spreadsheet creation as specified below).
C:\POM\CITMOD	This directory contains the application file CITMOD.EXE (the end-user model) which allows users to change tax rates, benefit amounts and save manifesto files.

Running the Model-Builder Application

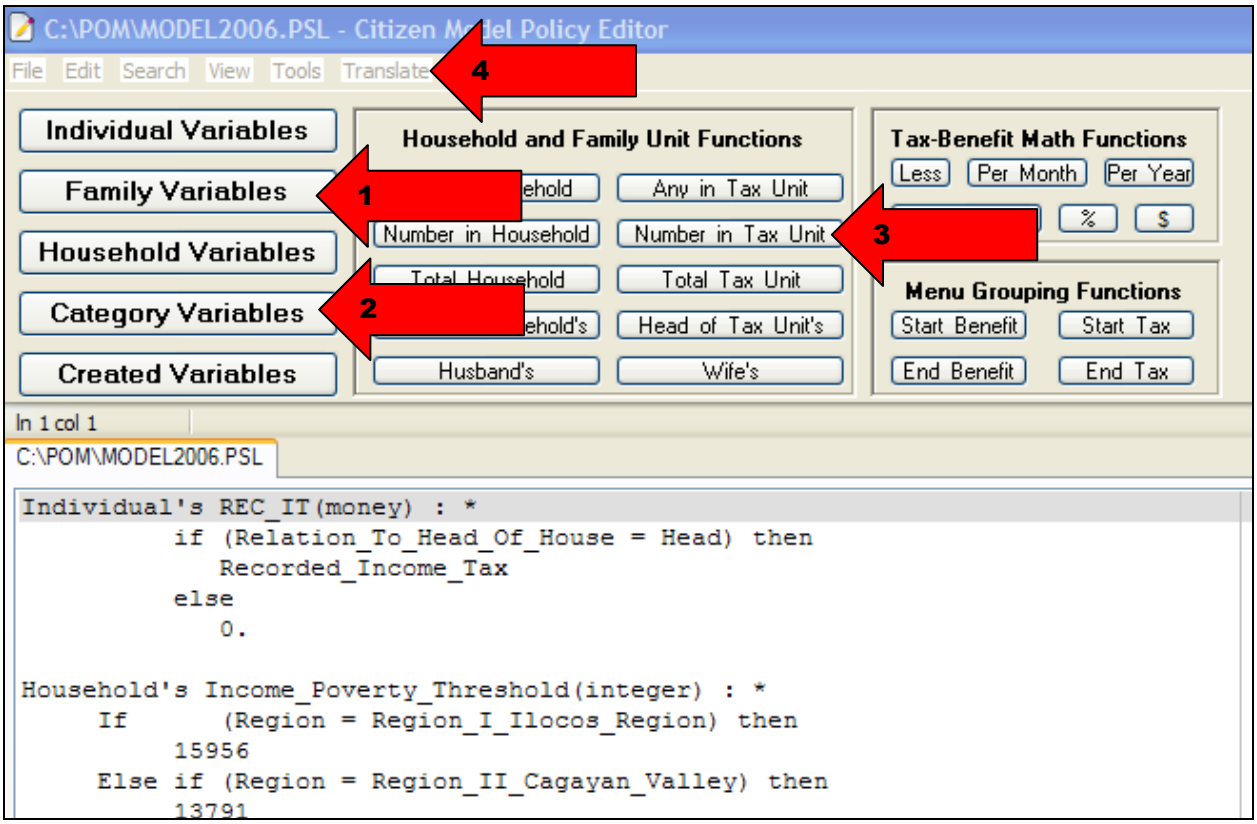
Execute the file \POM\MODBUILD\MODBUILD.EXE. Your screen should contain the image shown in Figure 15 below.

The arrows shown on Figure 15 show some special features of the Model Builder program as follows:

1. Clicking the buttons labeled "**Individual Variables**", "**Family Variables**" and "**Household Variables**" will cause popup menus of variable names from the income/expenditure survey to appear in a separate window. If the user double-clicks on the currently selected row then the full name (the long descriptive label) will be inserted into the text editor window at the bottom of the screen. If the descriptive label contains spaces underscore characters are used to create a long variable name without white space. The coding philosophy of PSL is to avoid short cryptic labels to ensure the code is self-documenting. For SPSS users this might be thought of as using VAR LABELS rather than VAR NAMES in the code.
2. Clicking the button labeled "**Category Variables**" will open up a popup menu listing all those variables associated with category lists. To see the list of categories for a particular variable double click that variable name. To insert the full descriptive label into the text window at the current cursor position then double-click the category name.
3. On the right hand side of the screen there are a range of buttons on panels labeled "**Household and Family Unit Functions**", "**Tax-Benefit Math Functions**", and "**Menu-Grouping Functions**." If you click any of these buttons that will cause the relevant PSL programming key word to be inserted into the text at the current cursor position.
4. On the pull-down menu bar the "**Translate**" button will cause your entire set of PSL commands to be translated into pascal source code in the form needed to create an End-User Model with no

further programming. You must ensure that the cursor is at the first character position *on the top line of your model* for the translation process to execute properly. If the translation process executes with no errors then CITMOD.exe can be built using the Delphi 7 compiler with no further programming.

Figure 15: Open Screen of MODBUILD.EXE

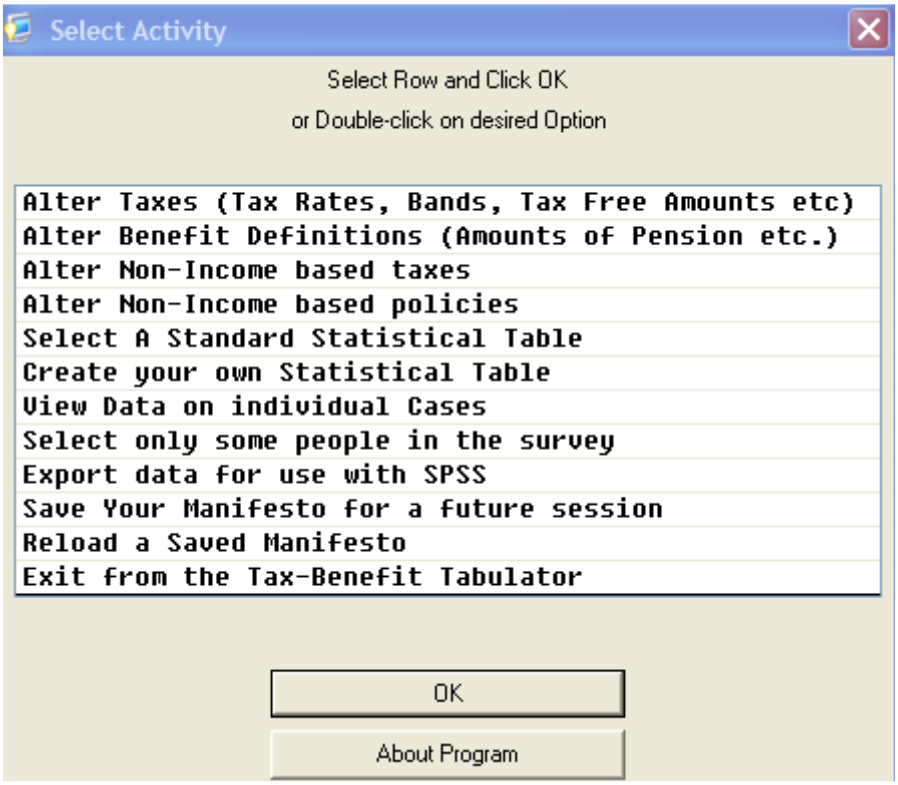


Running the End-User Model

After running the CITMOD_Intaller.exe program an example end-user model will be installed in the directory \POM\NEW. In this directory run the file CITMOD.exe. You will see a brief information screen explaining that this version of the program uses only fictitious data. To run the full model you would need to have a copy of the combined Philippine Family Income and Expenditure Survey / Labor Force Survey for 2006-07. This should be obtained from the Philippine National Statistical Office (WWW.NSO.GOV.PH) after providing proof of purchase of this data you should request a copy of the CITMOD formatted data (which recovers family units and sorts the data into individuals, family units and households). Send an email request to author@FilipinoFutures.com .

After the initial information screen you will see CITMOD's main menu as it appears in Figure 16 below. The principal difference between the range of options within MODBUILD.EXE and CITMOD.EXE are that CITMOD.EXE has no programming language interface. One can only change constants (such as tax rates and the amounts of tax allowances) within CITMOD.EXE. In MODBUILD.EXE one can design a completely new tax or benefit so long as it can be simulated from the variables in the income/expenditure survey.

Figure 16 Main menu options of the Citizen Model



The functions of the main menu options shown in Figure 16 are as follows:

1. **Alter Taxes**: This allows you to vary the rates of tax associated with a given tax and any associated tax allowances or maximum payments (Filipino Social Security is has a monthly maximum contribution). There are sub-menus for each tax. Several taxes try to simulate compliance rates and income reporting rates. In the current version of CITMOD a 'Compliance Rate' shows the proportion of those who should pay a given tax who are un-registered (that is who do not pay it at all). An 'income reporting rate' specifies for those who are registered what proportion of their income is reported.
2. **Alter Benefit Definitions**: These allow you to change the amounts and any tapers of a benefit.
3. **Alter Non-Income based Taxes**: Allow simulations of tax changes where there is no way of modeling it from the income/expenditure data (for example a change in corporation tax or import duties)
4. **Alter Non-Income based policies**: This refers to policies where it might be difficult to attribute a cash value to individuals. This must be associated with a unit cost (such as the cost per pupil of education or the cost per kilometer of train line).
5. **Select a Standard Statistical Table:** This offers two tables relating to the national budget. The standard revenue table shows the tax revenue or cost (before and after a policy change) of each tax or benefit simulated. A second table shows this information only for the taxes with their rates unchanged and compares the resulting tax revenue with tax revenue statistics taken from government publications. Several other tables show standard tables of the distributional effects of any policy changes made in the benefit / tax menus under headings 1 or 2 above.
6. **Create Your Own Statistical Table**: This allows you to build up your own table definition by selecting the row variable, column variables and one or two variables for the contents of each cell.

7. **View Data on Individual Cases**: This is usually only needed for debugging purposes. If some people unexpectedly lose money from a policy change when one expects them to gain this feature can allow the inspection of individual family variables as the tax/benefit is being calculated
8. **Select Only Some People in the Survey**: This is usually only needed in conjunction with the 'View Data' function above. If you want to find some people who are unexpected loser from a policy change you can define up to three Boolean conditions that will define which households will be displayed. This may also be used to limit the cases that are used to produce statistical tables. If a subset of cases is defined before running the statistical tables (for example only those with a head of household above the retirement age) then the table will be so constrained.
9. **Export Data for Use with SPSS**: This will read a list of variables defined in tax file to be exported for use in SPSS. This may include some of the original data variables and new variables calculated within CITMOD.exe
10. **Save your Manifesto for a Future Session**: This takes all your benefit and tax changes and saves them in CITMOD format.
11. **Re-load a saved manifesto**: This loads a manifesto of tax/benefit policy changes.

Preparing Software for Use with Delphi 7

In order to use the output from MODBUILD.EXE you must have a copy of Delphi 7 available. You should de-archive the source file archive called "CITMOD_source.rar." The project file is:

Modbuild.dpr

MODBUILD requires the following packages to be obtained as shown in table below.

Package and Location	Function and Availability
\Program Files\Berg\Next Suite\Sources; \Program Files\Berg\Next Suite\Sources\Next Grid; \Program Files\Berg\Next Suite\Sources\Next Addons; \Program Files\Berg\Next Suite\Sources\Next DBGrid; \Program Files\Berg\Next Suite\Sources\Next Inspector; \Program Files\Berg\Next Suite\Sources\Next Collection; \Program Files\Berg\Next Suite\Sources\Next Sheet; \Program Files\Berg\Next Suite\Sources\Next TBX;	These packages allow the creation of menus and Spreadsheet style output files Version 4.9 Should be used. Packages should be obtained from Bergsoft (www.Bergsoft.net)
\Program Files\APMap\Units\Delphi7	This should be installed if shaded maps are required. (amber.ivanovo.ru/**apmap**.htm)
C:\CitizenModel\Editors\eedit7\SynEdit; C:\CitizenModel\Editors\eedit7\SynEdit\Packages; C:\CitizenModel\Editors\eedit7\SynEdit\Source	This is a public domain package to produce syntax colored source code. Open Source – included with archive file.

Agent Based Modeling with CITMOD

To imagine how CITMOD might be used as a tool for agent based research one should consider two previous uses of the system:

Unemployment Research

In-work benefits such as Family Credit are designed to alter the replacement ratio: the ratio of incomes received by the unemployed on benefit compared with post-tax incomes in work. A probabilistic model may simulate the likelihood that given person will be employed based on a change in the replacement ratio resulting from a change to tax rates and benefit amounts (Truscott 1994).

Tax Evasion Research

To simulate the effect of a household evading a particular tax one may use the random() function within PSL to calibrate tax payments to published totals for tax-payers that appear in administrative data. For example, if the CITMOD predicts that there should be 10 million Income Tax payers but administrative data shows that only seven million actually pay the following equation would calibrate the model accordingly:

```
Income_Tax_Compliance_Rate = random(75).
```

A variable inside the brackets could be used to supply a proportion that would allocate different compliance rates to employees and the self-employed.

Index

References

1 Truscott, P.D., (1987) “Computer Models of Tax-Benefit Policy”, Guildford: UK, University of Surrey.
2 Truscott, P.D., (1994) “An End to Unemployment”, London: Movement for Christian Democracy.
3 Suplicy, E.M., (2008) “Basic Income and Employment in Brazil”, Louvain-la-Neuve, Belgium: Basic Income Earth Network.
4 Tutu, D., (2006) "Message to the 11th International Congress of the Basic Income Earth Network", Louvain-la-Neuve, Belgium: Basic Income Earth Network.
5 Alatas, S.H., (1999), "Corruption and the destiny of Asia", Malaysia : Prentice-Hall.
6 Keith Stallard has worked as a water supply consultant to the governments of Kenya, Vietnam, Papua New Guinea and the Peoples Republic of China. See Stallard, K., 2008, “Stallard and Associates: Achievements and Clients”, retrieved on October 20th 2008 from http://www.stallard.net.au/ache_clients.html
7 Matisonn, H., Seekings, J., (2002), “Welfare in Wonderland? The Politics of Basic Income Grant in South Africa, 1996-2002”, Louvain-la-Neuve, Belgium: Basic Income Earth Network.
8 WVS, (2006) “European And World Values Surveys Four-Wave Integrated Data File, 1981-2004”, v.20060423, 2006. Cologne, Germany: The European Values Study Foundation and World Values Survey Association.
9 Velasco, D., (2005), “Go! Young Progressives in Southeast Asia”, Friedrich Ebert Stiftung retrieved on October 2nd 2008 from http://www.fes.org.ph/pdf/yp/countrypapers_philippines.pdf
10 This is a personal observation of the proponent who has also been a student or a teacher in the UK, USA, Vietnam.