CeRP
CENTER FOR RESEARCH ON PENSIONS AND WELFARE POLICIES

Working Paper 85/09

THE LIFE-CYCLE INCOME ANALYSIS MODEL (LIAM): A STUDY OF A FLEXIBLE DYNAMIC MICROSIMULATION MODELLING COMPUTING FRAMEWORK

Cathal O'Donoghue
John Lennon
Stephen Hynes

April 2009

The Life-Cycle Income Analysis Model (LIAM): A Study of a Flexible Dynamic Microsimulation Modelling Computing Framework[1]

Cathal O'Donoghue , John Lennon, Stephen Hynes

Rural Economy Research Centre (RERC), Teagasc, Ireland

Abstract

This paper describes a flexible computing framework designed to create a dynamic microsimulation model, the Life-cycle Income Analysis Model (LIAM). The principle computing characteristics include the degree of modularisation, parameterisation, generalisation and robustness. The paper describes the decisions taken with regard to type of dynamic model used.

The LIAM framework has been used to create a number of different microsimulation models, including an Irish dynamic cohort model, a spatial dynamic microsimulation model for Ireland, an indirect tax and consumption model for EU15 as part of EUROMOD and a prototype EU dynamic population microsimulation model for 5 EU countries.

Keywords: Microsimulation, Computation**.**

1. Introduction

Population-based dynamic microsimulation models are programs that are used to forecast populations into the future and to assess the impact of economic and demographic change on public policy. In particular these models have been used to analyse existing policy and to design policy reforms in inter-temporal policies such as education, pensions, long-term care and spatial policy.

The objective of this modelling framework is to incorporate the time dimension into policy analysis. Using models based on cross-section data simply allows one to look at the effect of policy at one point in time. Using cross-sectional data one is limited in the simulation of policy instruments which depend on inter-temporal factors such as pensions. A dynamic microsimulation life cycle model allows one to examine policy over time; for example life course redistribution, forecasts of cross-sectional redistribution and the simulation of pensions. We describe an innovative computing framework used to create dynamic microsimulation models.

[1] Acknowledgements: The authors gratefully acknowledges financial assistance from the Postgraduate Fellowship of the Economic and Social Research Institute, Dublin, the Irish Department of Agriculture Rural Stimulus Fund, Center for Research on Pensions and Welfare Policies (CERP), University of Turin, the NUIG Millennium fund, the Combat Poverty Research Fund and the *Targeted Socio-Economic Research* programme of the European Commission (CT97-3060), the AIM project financed under the 6th Framework Research Program of the European Commission and the IDARI project financed under the 5th Framework Research Program of the European Commission. I am very grateful to the assistance of Donal Kelly who substantially improved the code of this model. I am grateful to comments provided by Geert Bryon, Jane Falkingham and seminar participants in LSE, Nordic Microsimulation Workshop Copenhagen, AIM workshop Madrid and colleagues in Teagasc, the DWP and NUIG. The author is responsible for all remaining errors.

Designing dynamic microsimulation models is a large model building project involving many disciplines such as economics, social policy, statistics and computer science. This paper describes a mechanism for making dynamic microsimulation models easier to construct, using a generalised method. In particular we describe a number of applications of LIAM.

The paper is designed as follows. Section 2 describes the objectives of the paper, with section 3 describing the main model features. Section 4 overviews the different types of process modules, while section 5 discusses alignment. Section 6 discusses some efficiency features. Section 7 describes some of the implementations of LIAM and section 8 concludes.

2. Objectives

The construction of a dynamic model is a very large task, both in terms of grasping the types and forms of behaviour that take place over a lifetime and the effort in programming 1000's of lines of code.

Despite dynamic microsimulation modelling (DMM) as a science having existed since the 1970's (see Orcutt et al.), the field has progressed only slightly (See O'Donoghue 2001). Part of the reason has been the resources requirements. When DMM's were first developed, they were in fact advances in computer science as well as being advances in social science methodology. Likewise in many countries, data limitations have prevented the development.

However in recent years, both difficulties have been overcome as computers have increased in speed and thus allowing for very powerful models to be constructed on PC's. The establishment of household panel datasets in many countries, for example the European Community Household Panel Survey, the British Household Panel Survey and the German Socio-Economic Panel and the increasing availability of administrative datasets has removed the barrier to the estimation of dynamic behavioural processes.

However despite these advances, the spread of the DMM technology and the development of the field has been relatively slow. The models that are being used at present are not doing very much more than the DYNASIM model in the late 1970's. A large potential reason is the apparent benefit to cost ratio. Many institutions when faced with the large cost of developing a dynamic model felt the money better spent on other techniques.

One significant contributor to the cost of development is the cost in actually producing the computing environment of the model. Because the computing necessary to produce a dynamic microsimulation model is so complicated, computing development has often taken precedence over developing better behavioural equations. It is therefore important to focus on ways of reducing the cost of building this initial framework.

O'Donoghue (2001) surveys the dynamic microsimulation models that have been constructed, describing in particular the design choices that have been faced. While most models have been built as stand-alone efforts, a number of attempts have been made to avoid the start-up costs and learning curve in building the model by utilising

the same framework for alternative applications. There were some efforts in the 1970's to write actual microsimulation computer software packages. However because of the complexity of the system to be simulated, users have more specialist requirements than these software packages allowed.[2] Although not designed with objective of constructing multiple dynamic microsimulation models, the code from CORSIM model (See Caldwell, 1996) has been stripped down and used as a template in the construction of the Canadian DYNACAN, Swedish SVERIGE and for the US Social Security Administration models. There have been four examples of programs that have been written explicitly for multiple Dynamic microsimulation model construction, ModGen (Wolfson and Rowe, 1998), UMDBS (Sauerbier, 2002), GENESIS (Edwards, 2004) and LIAM described here. ModGen is a computer language designed for create microsimulation models and has been used to create a number of microsimulation models (dynamic and static) within the Canadian government such as Lifepaths. MODGEN is an open model while LIAM is a closed model and the spouses in LIAM come from within the dataset. UMDBS is a simulation system developed at Darmstadt University as part of academic research. It is implemented in the object oriented language Smalltalk and its main applications are socio-economic investigations. GENESIS is a SAS based modelling framework being used within the UK Department of Work and Pensions to create the Pensim2 pension age dynamic microsimulation model and the state pension forecasting model. While LIAM is fully accessible to researchers, GENESIS is an internal government model currently not accessible to researchers.

In this paper we describe the LIAM framework, which was developed ironically because of the low resources available to the author. Dynamic models have typically been constructed by governmental institutions (MOSART, SESIM, DYNACAN, PENSIM2) or by major research grants (SVERIGE, DYNASIM, POPSIM), although a number of models have been constructed as part of PhDs (Harding, Baldini). Not being funded by a major research grant or by a government institution LIAM falls into the latter "low budget" category, being developed initially as part of a PhD and latterly expanded with small research grants and with the assistance of a number of PhD students. As a result, LIAM has necessarily to be less ambitious at the outset. However despite these shortcomings, it is hoped that with improved data and funding availability that LIAM can be improved in the future. Therefore the objective of LIAM is to construct a program which although relatively basic initially is not constrained from adapted for future uses, essentially has been future proofed to allow for future enhancements.

While the initial application of LIAM was to develop a single cohort analysis of life-course the redistributive impact of the Irish Tax-Benefit system using relatively unsophisticated data and behavioural equations a number of subsequent developments (described in more detail below) have occurred such as improved data (2-8 years in the panel data underlying the behavioural estimations), the addition of a graphical user interface, the move to a multi-cohort population model, international comparisons and the use for alternative policy analyses such as spatial and indirect taxation. Future developments that are planned include improving the behavioural equations to respond to changes in the policy environment such as labour supply retirement and migration.

[2] See Leombruni and Richiardi, 2005 (2005) for the development of a model using Agent Based Modelling

It is not possible to foresee the problems involved in developments in these areas in advance. However in order not to allow current limitations to inhibit future developments of the program, careful thought is necessary in the design of a flexible modelling framework. There are a number of features that would be desirable in such a framework to be able to meet these objectives in the future.

- In order to be able to deal with new datasets with ease, using different sets of variables should not be a problem.

- It should be easy to incorporate new behavioural information in the framework.

- Need to be able to run on a personal computer using standard "inexpensive" software.

- It should be straightforward to make changes to model using the framework even if the model has not been used for a period of time or is to be used by multiple analysts. This implies transparency in the operation of the framework and also flexibility in the way in which behaviour can be incorporated in the model.

- These points also imply that the framework should be robust to changes desired.

- Speed may not be considered a priority initially, but despite computing time decreasing with the availability of cheaper and faster computers, demand has increased at a faster rate. Efficiency improvement is discussed below.

- The objective of this modelling framework is to allow the user to focus more on the estimation of behavioural equations rather than computing issues.

- The modelling framework should also allow feedback effects of policy reforms to be examined.

3. Framework Features

In this section we describe the main aspects of LIAM, focusing initially on the general structure of the framework and then elaborating the data structure and issues relating to modularisation and parameterisation.

Structure of Framework

A dynamic microsimulation model is essentially a model that takes individual objects (individuals, households, farms, companies) and simulates the probabilities of various events occurring at various points in time.[3] Figure 1 describes the main operations of the ageing component of dynamic microsimulation model. Here the operation of one particular ageing module at one point in time is examined. In the model itself, this process would occur on a number of occasions as all the individuals in the database would pass through many ageing modules at each point in time.

[3] Dynamic events may of course occur at the same point in time as other events

Data for each person are firstly taken from the database having been transformed into the model data-structure, which is described in more detail below. The individual is then passed through each ageing module in turn. The ageing modules to be used are specified as part of a parameter list, which allows the order and the types of the transition processes to be varied. Input parameters for each ageing module are stored in Microsoft EXCEL spreadsheets (XL) and are accessible by the front end. Output from each ageing module is stored in alignment storage matrices in memory. For example, alignment regressions produce a deterministic component XB to which is added a stochastic component ε. These are stored in a dynamic data structure and ranked with the highest Z percent of values taken from the exogenous totals in the alignment process. If the ageing module is a transition between states, then the output will be a probability, otherwise if the ageing module is a transition between continuous amounts like for example incomes, the output is a real variable. When all individuals have been passed through the particular ageing module, alignment occurs (see section 5 below for a description). This ensures that aggregates from the micro model match macro aggregates. Finally if a variable for any individual changes then this change is registered in the database[4]. The rest of the paper will describe in more detail the operations of each of the components described here.

Data and Framework Data Structure

In this section we describe how data is handled in LIAM. We describe the database used, how data is stored within the framework, the data structure. The data structure or format of the data is very important as it determines to a large extent the amount of memory required to store the data, which in turn influences the speed at which the model can run. It also has important consequences for the flexibility of the model.

Turning first to data storage, we adopt a relational database structure due to organisation and memory handling advantages. Input data and outputted data is stored in ASCII format.

Figure 2 describes the data-structure used by LIAM. Structurally the data is stored in a hierarchy of object types (tobjt) such as person, household, firm etc.[5] Each of these object types themselves consists of a number of objects (tobj) such as the actual incidence of a person or household. Events (tvar1) such as births, tenure status or identification number then occur to objects. Each event can have a number of incidences or values (tval).

We exploit the hierarchical nature of relational databases making data storage event driven. Storing model output as consecutive cross-sections would result in severe inefficiencies, as each variable would be stored for each output period, so for example the gender would be stored for each point in time. Making data storage event driven, new data is stored only when a new event occurs and thus the data changes. Gender is therefore only stored at birth. One can make significant savings in memory as a result. Each individual variable however requires more information than in the case of the

[4] Note by the term database, we refer to both the physical relational database, stored on the hard disk in ASCII and the virtual database we create in memory.

[5] In cross-sectional data structures, persons are considered at a sub-level to households. However because persons can be members of a number of different households over time, this relationship breaks down. In the data-structure persons are considered one set of objects and households another, where the ID's of the member individuals of a household are events that can occur to households.

cross-section data structure. For each event it is necessary to know what event occurred (tvar1), when it occurred (tval.evtime), and the value of the event (tval.amount).[6]

There are a number of ways in which data can be stored within LIAM itself during the simulation. If the model were open as in the case of the DEMOGEN or LifePaths models in Canada where new spouses are generated synthetically when needed, then all of each individual's transitions could be simulated independently of other individuals. Thus each individual could be read from the database, simulate their life course and store in the database one at a time. LIAM however uses a closed methodology where individual behaviour can be dependent on the characteristics and behaviour of other members of the sample. Alignment is an example, whereby the employment of an individual is not independent of the rest of the population, but depends upon the interaction of the market. Utilising a closed model means that except for new births or immigrants, no new individuals are generated. Marriages for example link individuals already in the database. This method is more straightforward to interpret as it mirrors the actual population. The model is not solely individual based (it is a multilevel model – Regions, Counties, Districts, Households and Person) as many operations in the model depend on other individuals such as the marriage market, processes which depend on spousal information and alignment routines (see below for a description). A side effect of this is that it is necessary to store all individuals in memory during the simulation. The virtual database stored in memory during the operation of the program mimics the structure of the relational database stored on the hard disk.

Once the data has been read from the database into memory, LIAM runs through each object type (person, family etc.) in turn simulating the life course events desired for each object of that type. Simulation processes are therefore object type specific.

Typically variables which are components of the household data structure are declared in long lists within a dynamic model. They may be initialised elsewhere and have other operations carried out in other parts of the program. As the modelling framework is so large and complicated, it may be difficult to keep track of all the places in LIAM which need to be altered when a new variable is included. Therefore, in order to keep the framework flexible and yet maintain the robustness, it is desirable that the number of alterations necessary is kept to a minimum. As a result instead of declaring variables within LIAM, we declare the list of variables to be used separately in a parameter sheet (dyvardesc).LIAM then creates space for the variable, initialises the data and carries out all necessary transformations and operations automatically and therefore is entirely flexible with regard to the set of variables used within the framework. Thus if the user wishes to introduce a new instrument with an output variable such as health status, then the user simply needs to introduce the variable into the parameter sheet and LIAM will do the necessary steps, without having to recode the framework. Another advantage of the flexible declaration of variables described is that because variables are stored in vectors, new composite variables can be produced easily. For example, a complex variable like disposable income which is not simulated directly can be generated from the vector of its components such as employment income and capital income.

[6] Another means of reduce storage space is to store variables as integers rather than as real numbers. Therefore when storing output, variables are first multiplied by 100 and then truncated.

Another important advantage of the hierarchical method of data storage is the ease in which duration information can be accessed. As the date and value of each event is stored it is possible to determine such information as duration, duration in the last 12 months, date an event first occurred, date an event ended, duration in a particular state and so on. Information of this kind is frequently required by tax-benefit systems and other policy analysis. Additionally it is easy to access previous values of an event such as previous earnings etc.

Population versus Cohort

The initial database depends on the purpose of the simulation. One of the key differences in the literature is the distinction between longitudinal or single cohort models and population or cross-section/multi-cohort models. However, this distinction is now largely redundant due to advances in computing power. From a computing perspective, a cohort can simply be seen as an initial sample of unrelated individuals aged 0, while the population contains a sample of individuals of different ages, some of whom are related. As a result, a decision about this does not have to be made about this, as the computing framework has been developed to handle both types of analysis. Running LIAM as a dynamic population model requires that the initial cross-section is stored in the required manner, while running the model as a dynamic cohort model requires the model to first generate an initial cohort.

Modularisation

The use of modularisation is an important technique that helps achieve the objectives of flexibility, transparency and robustness that LIAM requires. Modularisation means that components within the LIAM are designed to be as autonomous as possible. Modules are the components where calculations take place, each with its own parameters, variable definitions and self contained structure with fixed inputs and outputs. The result is a set of independent components that do not interact with each other directly, allowing the framework to operate as a collection of independent building blocks. Because each process module is entirely self contained, each can be run independently, left out or new modules included. Constructing a program in this way allows for the model to be easily expanded to deal with new behavioural equations or functions. Also because it allows the user to focus on individual components one at a time, without interaction with the rest of the program, the model becomes more robust.

Linkages

Many policy instruments depend upon multiple units of analysis. So for example, pensions may depend upon individual characteristics such as contribution histories, age etc., taxation may depend upon family characteristics such as both spouse's incomes and social protection instruments and welfare measures on the household unit. Similarly sociological analyses and long-term care analysis may depend upon wider kinship networks. These linkages are not strictly hierarchical (e.g. region, household, family, individual), they may in fact consist of a web of linkages (e.g. region, firm, household, family, individual, mother, father, partner, children etc.). This multi-level structure with its complex interactions between levels is one of the main complications of microsimulation models that make it difficult to use person-based modelling frameworks. While it is not infeasible to simulate using non-hierarchical linkages such as those between relatives across households using other software

packages such as social simulation and statistical packages, the non-hierarchical structure often requires one to be "creative" in designing the model due to inflexibilities in the model as they are often non-standard requirements. However specially designed microsimulation packages such as LIAM can have these data structures in-built improving the transparency and flexibility of the modelling environment.

In the LIAM framework, the mechanism of linking objects has been automated as a relational database. Potentially any object can be linked to an object of either the same or different object type. For example individuals of object type person (p) will be linked to their household of object type (h), while in turn the household is linked with the individuals in the household. So therefore to create the number of persons in a household, this process is carried out at the household level. This new household level variable npers can be accessed by individual processes using the prefix h_npers. Similarly we can have linkages between objects of the same type, accessing a mother's information, say education level using m_edlevel or father's f_edlevel.[7]

In the initial framework, there are no predefined linkages as the objects can be of any type defined by the user. The user pre-defines all linkages using the parameterisation described below to essentially create a web of linkages between objects; essentially defining keys to link tables. As long as the nature of the linkage is defined, it is then possible at any level of the model to access information from another level. This is quite a powerful feature of the data-structure, saving both time and memory. In the absence of these linkages, a h_npers, a new process would have to be simulated which would store this variable as a person level variable p_hnpers (say), which is analgous to a flat file, where household level variables are stored at the person level. The use of linkages or keys provides the space saving advantages of a relational database and avoids the simulation of an extra process to convert the household variable to the person level.

New Objects and Killing Objects

While creating a new object (person, family, household, enterprise) is itself not a very complicated task, creating space and assigning initial default values, creating a new object to mimic the birth of a new person is rather more complicated. As a result we have had to develop a specific as opposed to generic new_birth function. The assignment of variable values such as single, age zero, no education etc is straightforward. However it is also desirable that the new child inherits the linkages of the parent. So for example the partner (if any) of the mother at birth becomes the father, similarly the children of the mother become siblings and the hierarchical linkages such as household, family, region etc of the mother are also inherited by the child.

Analogously, "killing" a person is also more difficult that killing an object. The individual needs to be extracted from the web of kinship networks and other linkages of which they form part. Similarly the process is not independent of the characteristics of other objects. The number of persons in a household decreases by one, the spouse becomes a widow. Bequest of accumulated wealth may need to be transferred and in the case of pensions systems, contributions or entitlements may need to be transferred

[7] Prefixes are defined by the user.

to surviving dependants. Again the possible complexity of the operation is far too difficult to generalise and so again a specific routine has been required

Migration

Migration is another complex operation. From the point of view of LIAM, emigration is analogous to killing someone as in a national model, we do not track individuals while they are abroad.[8] A variation of the pageant algorithm (See Chénard, 2000) is used to ensure that the migration of family units results in national individual level aggregates being achieved.

Immigration however is a more difficult situation. An immigrant differs from a new birth in that they have an accumulated set of characteristics and potentially is accompanied by other family members. One solution is to simulate the range of characteristics of new immigrants on arrival. However the range of characteristics is very broad and variable and so it would be very difficult to retain the correct multi-dimensional distribution of characteristics. To avoid this problem we sample (with replacement) from a set of immigrant households in the data. Thus whenever we need an immigrant household we simply select a "real" (data-dependent) family with actual characteristics of a new immigrant family. In addition to preserving the multi-dimensional distribution, it saves substantial computing time in having to simulate all the individual characteristics.

Parameterisation

In order that modules and other components of LIAM can be changed with ease, it is necessary to store model parameters externally. So where possible no parameters are hard coded within the framework. Figure 3 details the set of parameters used by the modelling framework. The sets of parameters, representing the *flow of control* in the model, are in some sense hierarchical.

At the top level we have *dyrunset* parameters which contain the parameters necessary to run the model, detailing directories (location of input and output files), time period to be run etc. Figure 3 is divided in two by dashed lines indicating sets of parameters dealing with the data structure and sets of parameters dealing with the simulation process.

On the data side the highest level parameters are contained in *objtype*. This file tells the model how many object types there are (region, household, person etc.). LIAM creates each object type based upon the list defined here and assigns defined prefixes (r,h,p etc.). The framework then looks for files *objtype_x* containing the incidences of each of the object types (r, h, p, etc.) which in turn contains in the identification numbers or id's of each object of that particular object type. So objtype_p would contain the set of id's of all persons.

Related to the set of object types is a set of variables associated with each type in the *dyvardesc* file.[9] In this file, all variables used in LIAM are declared and described. It

[8] Technically as pension rights can be accumulated overseas, one may need to simulate their life-histories while overseas, but this has been beyond the capacity of any existing model.
[9] In the front end, the parameter files have equivalent menus.

is in essence a data dictionary for the model. This file contains information on the following attributes of each variable:

- variable name
- variable type (binary, multi-category, continuous)
- an income variable (monetary amount that can be added to other monetary amounts, for example it prevents adding gender to employment)
- limits of the variable (upper and lower bounds) for debugging and validation
- a categorical variables (if so how many categories and the list of categories – for tabulation purposes),
- is to be updated during the simulation (to account for inflation)
- default values to be taken by new persons
- data description (describes variable names)

Associated with each variable, there is a data table containing information about object associated with the variable (who), the time the event occurred (when) and the value of the variable (what).

While each data table within an object type is linked by the key or object ID, we need further information to link objects of the same or other type. The *linkage* parameters define the set of possible links between objects. The user needs to define a name for the linkage (ph – person to household, hp – household to person etc.) and equivalent origin (p for link ph) and destination (h for link ph) types. These linkages between objects are stored in the linkage file *link_xy*. Subsequently, for each linkage listed in link_xy, LIAM pairs the relevant origin objects (e.g. children) and destination objects (e.g. parents) and stores the resulting origin and destination IDs in a link-specific (e.g. pc) file.

We now consider the set of parameters that define the simulation processes. The highest level is the *agespine or* process spine/list. In any simulation there is an implicit ordering, and events are triggered through conditions. The process spine contains the list of modules to be run in the dynamic model, so that by varying the order of the modules and varying the content of the list, one can vary the types of processes that can be run in the model. This feature exploits the modularisation, where because each process is seen as a separate building block, the number, type and order of processes can vary without having to change the code. Each process or module has a corresponding parameter sheet in the parameter file "Transitions". These parameter files tell the model the output variables of each process, what type of process (described in the next section), whether a process needs to be aligned and the actual process parameters themselves such as the transition rates, regression equation and policy rules etc. If a particular process is to be aligned, then LIAM will look for an appropriate set of *alignment* parameters. Sometimes individual parameters may be required to be changed between runs without any change to the set of processes. A reform to a pension simulation module where the replacement rate was changed is an

example. The *polparam* parameter set contains information associated with each parameter for each possible "system", where the system to be run is defined in the *dyrunset* parameters.

4. Process Modules

This section describes the main process types that can be used by LIAM. This refers to the collection of operations that are simulated on objects during a simulation. These include demographic processes such as birth, marriage, having children and death, education, labour market processes such as employment and unemployment, the simulation of incomes and interactions with the tax-benefit system.

In order to aid flexibility, we classify processes under a number of headings. In this way, instead of programming each module separately, we only need to program the module type once. In order to run a module, we then only need a module name (which is included in the process spine), a module type to determine which program to run and a set of parameters which is fixed for every process type. At present there are 6 module types:

- transition matrices, in the form of a log linear model (*trap*)
- transformations (*tran*)
- regressions, both with continuous and limited dependent variable (*regr*)
- macro alignment (discussed in section 5) (*macro*)
- marriage market (*mmkt*)
- tax-benefit system[10] (*tb*)

The first component of a parameter file contains details about what conditions need to hold for the process to be run. At each point in time, each individual is passed through the module. If the conditions hold, then the module calculations are carried out and the output passed to the alignment component of the module. The output for each individual is stored until all individuals have passed through the module. The alignment component then ensures that the aggregates correspond with external control totals.

Transition Matrices

One of the most important processes in a dynamic model is the transition between different discrete states. Transition Matrices are often used to perform these operations. They specify the probability for an individual of particular circumstances to move from state A to state B. In this framework, transition matrices can be stored as log-linear models (See Dobson, 1990). In this way transition rates are decomposed into average and relative transition rates. In this way extra-relative transition rates can

[10] The tax-benefit system is in fact a collection of modules. We have linked the dynamic model to the EUROMOD EU15 tax-benefit model and to other tax-benefit routines.

be added with ease. For example, if a mortality rate on average fell by 0.1% every 10 years, then a relative probability time dependent parameter 0f 0.999 could be added. Similarly it also allows the model builder to combine information from different sources. So for example we combine actual age-gender specific mortality rates for 1991 taken from life-tables and use relative mortality rates taken from (Nolan, 1990) that incorporate socio-economic relative mortality rates.

Regressions

The second type of transition process used are those based upon standard regression models. At present, this type of module allows four types of dependent variable

- standard continuous dependent variable
- log dependent variable, allowing for use of the log normal distribution.
- logit discrete choice dependent variable
- probit discrete choice dependent variable

Any variable in the model can be used as a dependent variable and any variable can be used as an explanatory variable. The error term can also vary. The default error term takes a normal distribution with independent disturbances. LIAM also allows for the error term to be decomposed into individual specific (u_n) random effects and general error components (v_{nt}) (See Pudney 1992). However more complicated error decompositions are also possible. This allows some degree of heterogeneity to be assigned specifically to individuals. So for example in determining earnings, the individual specific error may represent some difference in innate ability, while the general error term represents random variation over time. Breaking up the variation in this manner will tend to reduce within lifetime variation and prevent to some degree the existence of very unusual life paths.

In this framework, transitions occur at discrete time intervals because of the weakness of the data and because of the desire to be able to align the data.[11] As Galler (1997) points out some statistical difficulties relating to the use of discrete time models, it is desirable to use short term discrete time periods such as a month. As the computing requirements can be substantial for monthly simulations, LIAM is sufficiently flexible to allow the user as the ability to specify the time period to be used and so monthly or annual periods can be simulated.

Transformations

While regression models and transition matrices are stochastic processes, involving a random component, some processes are deterministic. Examples include age, which depends on the date of birth, widowhood, which depends on the death of a spouse and so on. Likewise if an individual moves from year 6 in education to year 7, years of

[11] O'Donoghue (2001) describes some of the advantages and disadvantages of continuous time versus discrete time.

education increase by 1. This component has also been parameterised as transformations.

Within the transformations there are two types of deterministic transformation *gen* and *fgen*. The *gen* functions are of simpler types, utilising a calculation routines combining sets of variables using standard operations.[12] The *fgen* set of functions is where we program ad-hoc programs. It is where we for example we exploit the relational database structure of the data in operations such as the number of persons in a household, where the function counts the number of objects of type person linked to the object household as defined by the link_hp link file. Similarly it is where ad-hoc functions such as *new_birth* and *killperson* are defined. While *gen* functions and predefined *fgen* functions are pre-coded and parameterised so that new users can employ them, new *fgen* functions such as the pension system of a new country need to be programmed by the user if the existing functionality does not allow it.

Marriage Market

If an individual is selected to marry or form a partnership then, a process is needed to determine which spouse they will take. The process used here is to take the characteristics of the individual chosen to marry and the characteristics of each possible spouse and determine the likelihood of a match. Similar to the method used in other models such as the CORSIM model, this is done using a logit model that estimates the probability of marriage between pairs of individuals. The parameter file therefore is identical to that used in the regression process type. The module itself forms a matrix of the characteristics of the n men and n women selected to marry. Estimates a probability for each pair and assigns a match to the couples with the highest probability of marrying.

Bouffard et al (2001) has identified some problematical issues associated with the marriage market, in particular with strange matches occurring amongst the last people to be married in a particular simulation. In order to avoid these issues, we allow the user to create a super-set of potential male partners, so that rather the last female in the marriage pool to be select an unlikely match, there are a number of males to choose from. In addition we employ the Order of Decreasing Differences algorithm of Howard Redway at the DWP, which creates a measure of the distance of an individual from the centre of the population (or the average characteristic of the population) and selects the females with the most unusual characteristics, who are likely to be the most difficult to match, to be matched first. The logic is that those in the centre of the data “average people” are more likely to find a good match than someone at the extremes.

Policy Processes

The fourth process type is the simulation of the tax-benefit system. Here we describe how it is implemented in the program. Again, to re-emphasise the desire to reuse code wherever possible and to avoid duplication, the dynamic framework is flexible enough to link with other specialist programs such as tax-benefit models. Tax-benefit routines from other models such as EUROMOD can be seamlessly accessed by this model and thus can be used as module components of the dynamic model.

[12] {+,-,*,/, max,min,^,(,)}

Behavioural Response

A desirable feature often ignored in dynamic microsimulation models is the ability to include feedback loops so that behaviour can respond to changes in public policy. This is a criticism made by PRIM (1997), is that dynamic models are insufficiently flexible to incorporate the demands of behavioural response. In order to be able to simulate behaviour, typically the model needs to be able to call a policy simulation routine a number of times to quantify the financial impact of alternative choices on the decision in questions such as the choice to work or retire etc. In O'Donoghue (2000) we implemented a simple labour supply model where labour supply depend on tax-benefit policy, the tax-benefit system. The software framework has been designed to be able to incorporate feedback loops. The degree of modularisation that exists in the framework allows any number or order of modules to be run and for modules to be able to be run a number of times.

Thus for example in order to have labour supply depend on tax-benefit policy, the tax-benefit system will need to be run once as an input into the labour supply module and again once labour supply has been determined, taxes and benefits need to be calculated again on the resulting behavioural decision. In O'Donoghue (2000), the model used the tax-benefit system as an input into decisions to work, decisions to seek part-time employment versus full-time employment and to become self-employed. The tax-benefit system therefore needed to be run 5 times to examine the impact of the system on the choice faced by an individual. When there are more that 1 adult in the household, because behaviour of spouses can depend on each other, the tax-benefit system needs to be simulated 17 times (4 decisions for each, plus one run on the basis of resulting behaviour). As a result incorporating behavioural response can be computationally expensive.

Other possible behavioural routines that could be included are retirement decisions, consumption and benefit take-up. Although computationally expensive, the framework is sufficiently flexible should the user require and the computing power becomes available.

Robustness

Finally in order to avoid robustness problems due to modules being incorrectly specified, the model contains a debug device which ensures that all inputs required by a module are actually available (i.e. have either been generated in the model or read from the database) before each module can be run.

5. Alignment

The section describes the alignment function contained in LIAM. The objective of alignment is to ensure that output aggregates match external control totals. The reason this is done is that micro behaviour (both social and economic) is extremely complex and micro-theory being limited, cannot predict accurately all the variability of the system (in this case the life paths of individuals). In addition, a household model only makes forecasts about a small part of the economy and largely ignores interactions with the rest of the world economy. Also, data taken from relatively short periods of time may not fully reflect the dynamics within the household sector over time. As a

result dynamic micro-models may not be able forecast aggregate characteristics of the population well.

In the discrete choice models, the output for each individual is a probability. In order to use these models for predictive purposes, a decision rule is necessary. In other words, what forecasted probability or higher will produce an event. In order to predict a state with a logit (or probit model), one draws a random number uniformly distributed number u_i. When $u_i < logit^{-1}(\alpha + \beta X_i)$ (or $u_i < probit^{-1}(\alpha + \beta X_i)$), then a state is predicted to occur.

Another use of alignment is in correcting for predictive failures of econometric models. For example when using discrete choice models such as logit or probit models, often, the predictive power is poor. Duncan and Weeks, (2000) highlight that *"even in functionally well-specified models, the predictive performance is poor, particularly where some states are relatively densely or sparsely represented in the data"*.[13] Thus the further the probability of an event occurring is from 0.5, the less effective these decision rules are at producing the desired result. As a result models may under or over predict the number of events. So for example if 5% of individuals of individuals should have the event, then the logit model may not necessarily produce 5% of events. Alignment will however constrain the event to occur to 5% of individuals. This is effectively a calibration mechanism and will produce the correct proportion of events. Care must be however taken in its use as it may disguise errors in the model specification.

The types of control totals that would be used to align to include:

- The aggregate proportion/number in a state or moving between states.
- The average event value.
- The distribution of values.
- The average growth rate in the value of an event.

In this paper we shall deal specifically with the first type

A simple analogy about the relationship between alignment and the process modules is that the process modules such as logit models produce a ranking variable, while the alignment mechanism selects the number of transitions. For example, in our econometric model we may have an equation of the probability of dying as described in equation (1), that depends on age, gender and whether an individual is disabled or not. Assuming that disabled people have a higher mortality rate, then given the same age and gender and distribution, as expressed by the stochastic component ε_i, the mortality distribution for disabled people will be higher.

$$\begin{aligned} logistic(p_i) = \quad & \alpha + \beta_1 \times Disabled_i + \beta_2 \times Age_i + \\ & \beta_3 \times Gender_i + \beta_4 \times Disabled_i \times Age_i + \varepsilon_i \end{aligned} \tag{1}$$

[13] The reason for this according to Greene (1997) is that "the maximum likelihood estimator is not chosen to maximise a fitting criterion based on prediction of y, as it is in the classical regression (which maximises R^2). It is chosen to maximise the joint density of the observed dependent variable.

The deterministic component of the model will result in those with a higher risk, having a better chance of the event occurring, while the stochastic part will ensure that there is some variability (so that not only those with high risk are selected). This model therefore produces the risk of dying.

In order to select the number of people that die, we use the alignment probabilities. Firstly individuals are grouped into the appropriate age and gender groups. As everyone in the relevant group will have the same age, gender and occupation, they only differ by the deterministic component for disabled people $\beta_1 \times Disabled_i + \beta_4 \times Disabled_i \times Age_i$ and the stochastic component ε_i. The object then is to select to die, the people in the group with the highest probabilities of dying. As β_1 is positive, proportionally more disabled will die than non-disabled. As a result we see that the output of the model equation is used to rank the individuals to whom the event occurs, but to leave the decision to the alignment process.

Implementation

In this section we describe a practical method for ranking individuals for alignment. We take as our reference point a logistic model:

$$p_i = logit^{-1}(\alpha + \beta X_i + \varepsilon_i) \qquad (2)$$

Utilising the model $logistic(p^*_i) = \alpha + \beta X_i$ will result in those with the highest risk always being selected for the event. So for example in our example given above, the disabled, all other things being equal would be selected to have a die. In reality those with the highest risk will on average be selected more than those with lower risk, rather than simply selected those with the highest risk. As a result some variability needs to be introduced.

Models based on the CORSIM framework such as the DYNACAN model (See Chénard, 2000) utilise the following method. Firstly, predicted probability is produced using our econometric model: $p^*_i = logit^{-1}(\alpha + \beta X_i)$. Next, a random number u_i, is drawn taken from a uniform distribution, is subtracted from the predicted probability, p^*_i, to produce a ranking variable, $r_i = p^*_i - u_i$. This value is then used to rank individuals so that the top x% of values are selected.

A concern about this method is that the range of possible ranking values is not the same for each point. In other words, because the random number $u_i \in [0,1]$ is subtracted from the deterministically predicted, p^*_i, then the ranking value takes the range $r_i \in [-1,1]$. However the ranking value for each individual will only take a possible range $r_i \in [u_i - 1, u_i]$. So for example if p^*_i is small say = 0.1, the range of possible ranking values is [-0.9, 0.1]. At the other extreme if p^*_i is large say = 0.9, then the range of possible ranking values is [-0.1, 0.9]. Thus because there is only a small over lap for these extreme points, even if a very low random variable is selected, then an individual with a small p^*_i will have a very low chance of being selected.

Ideally the range of possible ranking values should be the same, so that for each individual, $r_i \in [a,b]$, with individuals with a low p^*_i being clustered towards the bottom and those with a high p^*_i being clustered towards the top.

We now consider an alternative method. This method takes a predicted logistic variable: $logit(p_i) = \alpha + \beta X_i$. Next, a random number is drawn taken from the logistic distribution ε_i. This is added to the prediction of the $logit(p_i) = \alpha + \beta X_i$ to produce $logit(p_i) + \varepsilon_i$. The resulting inverse logit, $p_i = logit^{-1}(\alpha + \beta X_i + \varepsilon_i)$ is then used to rank individuals and similarly the top x% of households are selected.

The rank produced by the two methods is not the same. The second method will be more likely to select cases at extreme points than the first, while first method will select more points with central values of p^*_i.

Macro Alignment.

There are a number of levels at which alignment can occur. At the lowest level, alignment refers to the decision rule used in a discrete choice model. At the next level, described above in our mortality example, which is called the meso-level, concerns the idea that the aggregates for particular groups (in this case gender, age and occupation) should match the external totals. Meso-level alignment and the use of alignment as a decision rule can however be combined into one stage.

Sometimes the desired targets are narrower than the alignment targets we use. In our mortality alignment example, we align mortality by age, gender and occupation. We include occupation in the alignment because the occupational structure is very important for other characteristics in the model. However if say one of the core targets in the model is to achieve the mortality distribution supplied by external sources such as official population projections, which may only be by age and gender, then our meso-alignment may produce different aggregates. This will happen if our underlying occupation distribution is different to the one implicit in the official forecasts. It may therefore be desirable to adjust the results again to achieve these targets. This process is known as macro alignment. In the application of the framework used in this thesis, an example of meso alignment is the simulation of transitions between employment states. Macro alignment is then used to constrain total employment rates. See Appendix 1 for the steps involved in the macro alignment process.

Behavioural Change.

Handling behavioural interactions in the model resulting from alternative scenarios is another issue one needs to consider when deciding on an alignment strategy.

One potential solution is to examine the average (pre-alignment) event value such as the average transition rate or average earnings in the baseline scenario with the average in the alternative scenario. One potential method is to increase alignment values by proportional difference. This is a method utilised in some dynamic models.

This however assumes that all processes are *unconstrained.* This may be the case for example with the mortality rate. One may expect that an exogenous increase in human capital will reduce total mortality rates and thus one can shift down in the alignment totals is appropriate.

Some processes face market or other institutional *constraints*, issues that are only partially simulated in the model. An example is in the labour market such as the case where there is a behavioural change in labour participation in response to a tax change. If labour supply increases, then wages would fall and employment increase. This is similar to shifting the alignment probabilities. However one would have to shift earnings as well. However due to rigidities in the labour market, this may not necessarily happen. Labour Demand may be fixed, in which cane we may just simply see that as more women supply labour, they simply replace people in the labour market who are less "employable". This is similar to not shifting alignment at all. In cases where there are market interactions such as this, it may be useful to incorporate a model of the market that would inform the response of alignment totals to economic and demographic totals.

At present the framework makes no explicit incorporation of behavioural change in the alignment structure. Future work on macro-micro linkages will attempt to address this.

6. Efficiency

In earlier versions of the framework, developments focused on functionality and not speed. So for each process the model passes through all the objects of the object type, checks to see if a condition is true and if true, performs the calculation $f(X\beta+\varepsilon_i)$ and if necessary uses alignment to produce the predicted value of the process. In this section we describe a number of efficiency improvements that have been made recently.

One of the speed advances relies on the fact that most processes are relatively stable and so do not change much year on year. Because of this, eligibility conditions are unlikely to change much year on year. For example for lone parent births, the model used to check to see if an object is a female, single person of child bearing age. It does this for each year and for every person and was therefore very inefficient. Gender doesn't change and so it is inefficient to check to see if a male can have a child, Marital status only change infrequently and so the condition does not need to be recalculated each year. Similarly the age range condition only changes twice over the lifetime. One immediate speed improvement was to calculate the conditions for all people in the first year of the simulation and only to recalculate the condition if an input variable to the condition changed.

The same is true when calculating regressions. Most regressions are of the form $f\left(\sum_{i,j} X_{ij}\beta_j+\varepsilon_i\right)=f(A+\varepsilon_i)$ Again by calculating the value of the expression in the first year, when X_{ij} changes to X_{ij}*, one only needs to apply the following transformation $f\left(A+-X_{ij}\beta_j+X^*_{ij}\beta_j+\varepsilon_i\right)$.

The same speed efficiencies can be found for transformations, alignment and tabulations. For example when aligning by age-group, sex and education level or creating output tables by these components, most of these categories do not change much if at all during the simulation. It is therefore computationally quite expensive to identify all say 20-30 year old males with university education each period of the simulation to perform an alignment. Rather by computing the group membership at the outset and only change group membership when a characteristic changes, we reduce significantly the computational costs.

These improvements were applied by creating a data structure that links every variable to the processes which utilise the variable. When the value of the variable changes, then the related conditions, regressions, transformations etc are updated.

Moving simulations from periodic to initialisation plus simulation only when inputs change transfers some of the computing cost from simulation period to the start. Initialisation becomes a good deal longer as rather than simulating equations where conditions are met, (e.g. only simulating work equation for those who are in-work the previous period), we must now simulate all equations (e.g. simulate the equations conditional on working and conditional on not working in the previous period). However as no alignment needs to occur at this stage as all that is being calculated is $\sum_{i,j} X_{ij}\beta_j$, there are significant economies of scale and much less looping through the data.

As development of the program occurred incrementally and different versions have been run on different machines and with different specifications, it has been quite difficult to gauge the impact of the speed improvements. However a conservative estimate is that the run time is 10% of what it was and potentially as low as 5%. So the qualitative conclusion is that the gains are highly significant.

A relatively minor speed improvement was to store variables in a static rather than in a dynamic data structure (i.e. as an array rather than a list). As the set of variables does not vary within a run of the model, there is no gain to using a dynamic list; instead a speed penalty is imposed as the list needs to be traversed to find a particular variable as opposed to simply using the array index to identify the variable.

While attention was paid in the original data-structure to the memory efficiency of the data-structure by storing only new events, little attention was paid to the space taken within this structure, which proved very costly. For example all incidences of values were stored as *doubles* or real numbers, even though the majority of variables were binary variables that could be stored as a *char*. To improve this we introduced a new category in the data dictionary which specified what type of variable to be used, so that binary variables only took up a fraction of the space required to store a *double*. Also we stored real numbers as integers multiplied by 1000 and categorical variables as integers, taking half the space of a double. We also conducted an audit of the entire data-structure stripping out as much superfluous memory requirements as possible. This has been particularly important allowing for much bigger datasets to be simulated on a laptop with limited RAM.

Something we have not explored yet is a further speed improvement that could be found by creating sub-sets of objects associated with each condition. At present, the

model needs to scroll down through the whole dataset for each process. If conditions are updated dynamically, then the sub-groups where the condition is true can be updated dynamically, resulting in calculations only taking place on the subset. When the condition changes then, this updates the set of objects where the condition is true. Therefore in doing simulations, the model will only simulate over the set of objects eligible to be simulated.

At present all processes are run in series. In other words each process is completely simulated before moving on to the next process. This requires a data pass for each process. This is necessary for processes that are aligned as the decision about who makes a transition will depend upon all objects and not on individual objects. However some processes such as transformations and unaligned processes do not need to be done serially. Efficiencies could be gained by simulating these processes in parallel. For example if age is simulated for an individual, then age squared and age band could be calculated using age as input for each individual before continuing on to the next individual cutting the number of data passes and improving the speed of the model. Other examples include the calculation of durations and lagged values of variables.

As always in microsimulation models, there is a trade-off between flexibility, complexity and performance. Parameterisation may sometimes result in enhanced complexity and thus reduced transparency and ease of use of the model. In the LIAM structure, this has been less of an issue as the parameterisation allows for the same code to be reused over and over without recoding, so to some extent improving the transparency. There are however some performance overheads noted elsewhere in the paper, where the degree of parameterisation and generalisation may increase the number of operations required and thus increase the time to run a simulation. However this must be weighed up against the complexity of creating a dynamic model without an existing framework.

7. Implementations of the Framework

In this section we describe a number of implementations of the LIAM framework. Thus far there have been four implementations of the model

a. Life-cycle redistribution in the Irish Tax-Benefit system - Irish Dynamic Cohort Microsimulation Model

b. Redistributive impact of Indirect Taxes in Europe – dynamic microsimulation of expenditures in the EUROMOD framework.

c. Spatial Policy Analyses – Simulation Model of the Irish Local Economy (SMILE)

d. Cross-national comparisons of the distributional impact of pensions and the incentive to retire – multi-country dynamic microsimulation model – EU 6th Framework project, Old Age Income Maintenance Policies (AIM).

Model (a) was the basis of the author's PhD and has been used to examine the life-course redistributive impact of the Irish Tax-Benefit System (O'Donoghue, 2002) and the redistributive impact of pension reform (O'Donoghue, 2005). The implementation

was a single cohort model taking 1000 people aged 0 and simulating the entire life-history and then linking with the EUROMOD tax-benefit model to simulate the tax-benefit system. A feedback loop was used to incorporate the impact of tax-benefit policy on labour supply decisions.

While model (a) utilised an external tax-benefit microsimulation model to provide tax-benefit simulations for use in the dynamic microsimulation model, model (b) takes inputs from the dynamic microsimulation framework into a static tax-benefit model. As part of the EU tax-benefit model EUROMOD, there was a desire to examine the impact of indirect taxation on redistribution (See O'Donoghue et al, 2004). However most of the databases used as input into the model did not contain expenditure information. The LIAM framework was used to simulate a system of equations simulating total expenditure and budget shares of 20 groups of goods on the basis of information contained in the income surveys used in the model. Indirect taxes were then simulated using the EUROMOD framework. This model used datasets of up to 50000 households simulating indirect taxes for one fiscal year.

In recent years parallel microsimulation modelling has been used for geographical and spatial analysis (See Clarke, 1996 and Holm et al, 1996). Since 2002, a team comprising the University of Leeds, NUI Galway and Teagasc have been developing a model (c), the Simulation Model of the Irish Local Economy (SMILE) using the LIAM framework with the principle objective of carrying out spatial analysis in Ireland (See O'Donoghue et al, 2005). Examples include modelling the impact of local area demographic changes on welfare, modelling the spatial impact of rural policy reforms, identifying agri-tourism hotspots and eventually modelling the spatial behavioural impact of public infrastructure developments such as road building programs. The first component of the model is developed outside the framework, requiring the statistical matching of individual tabular local area census information with micro-level household data to produce the base dataset. This done using a statistical matching algorithm. The LIAM framework is used to simulate typical dynamic microsimulation variables such as demographic and labour market variables. Particular advancements from this model include regional labour markets, micro-farm level production functions and spatial behavioural models. The model is currently under development. This model, although divided into around 30 county models of about 70000 persons each simulates spatial based policy at the local level.

The fourth implementation of LIAM has just begun, where it is being used to carry out cross-national comparisons of the distributional impact of pensions in a selection of countries in the EU (Be, Ge, Ir, It and Sw). In addition a comparative analysis will be carried using a semi-structural retirement decision module based upon discounted income and pension wealth streams. This model simulates over a 50 year horizon 2000-2050, cross-sections of about 10-15000 individuals.

8. Conclusions

To conclude we have discussed some of the methodological innovations developed by the LIAM dynamic microsimulation framework. In summary some of the main issues are summarised in the following paragraphs.

Parameterisation has been used extensively throughout the model. This aids flexibility as code does not need to be reprogrammed when parameters change. This in turn improves the durability of the model as it allows new parameters to be included when better information becomes available.

Defining the data structure outside the model improves the transparency and the robustness of the model. When adding new variables to the model, alterations need only to be made in one place, in a parameter file. It therefore reduces the possibility of error and makes the model easier to change.

Using modularisation, all modules work independently of others which means that new modules can be added without affecting the integrity of the model. It therefore adds to the robustness of the model. Also, by allowing the user to focus on small sections of code at time, improves the transparency of the model.

Generalisation of main features of the dynamic model allows for the code which runs transitions, alignment and transformations to be reused for different purposes. Taking these as templates, one can declare a new module in the parameterisation of an existing type and simply change the parameters in order to produce a new process module. Also because the number order and type of module is parameterised, the model can handle any number of modules of each type and in any order without any need for extra programming. This is perhaps the most important feature of the model as it allows the model to be used for a wide variety of purposes. It thus allows for ease of expansion as improved data and micro-behaviour become available. Allow this not an attempt at writing a microsimulation programming language, it has allowed for a variety of different applications to be constructed without the need for extensive recoding. In addition it has been possible to use this framework as a template for other dynamic models because the model itself is entirely independent of data and behavioural equations to be used.

Lastly we have described a range of efficiency improvements in both speed and memory usage in developing the framework. While there have been substantial numbers of papers describing analyses carried out by these models, relatively little has been written on the technical development of the models. We do not claim to be the fastest or most efficient dynamic model; however we have attempted to document some of the issues that have arisen in the creation of this framework, so that hopefully other model builders can learn from the development process of others. To promote collaboration and further development, this model is available on request to researchers.

References

Atkinson, A.B. and J. Micklewright, 1985, *Unemployment benefits and unemployment duration*, STICERD Occasional Paper, No. 5, London: London School of Economics.

O'Donoghue, C., M., Baldini, and D. Mantovani, 2004, "Modelling the Redistributive Impact of Indirect Taxes in Europe: An Application of EUROMOD" *EUROMOD Working Paper no. 7/01*.

Bouffard, N., R. Easther, T. Johnson, R.J. Morrison, J Vink, 2001 "Matchmaker, Matchmaker, Make Me a Match" *Brazilian Electronic Journal of Economics*.

Caldwell S.B., 1996, "Health, Wealth, Pensions and Life Paths: The CORSIM Dynamic Microsimulation Model", in Harding A. (ed.) Microsimulation and Public Policy, Amsterdam: Elsevier.

Chénard D. 2000. "Individual alignment and group processing: an application to migration processes in DYNACAN", in Mitton et al. (eds.) Microsimulation in the New Millennium, Cambridge: Cambridge University Press.

Clarke, G. P. (1996) (ed.), *Microsimulation for Urban and Regional Policy Analysis*, Pion

Dobson A., 1990. *Generalised Linear Models.* London: Chapman and Hall.

Duncan A. and M. Weeks, 2000. Simulating Transitions Using Discrete Choice Models, in Mitton, L., H. Sutherland and M. Weeks (eds.), *Microsimulation Modelling for Policy Analysis: Challenges and Innovations*. Cambridge: Cambridge University Press.

Edwards S., 2004. GENESIS: SAS based computing environment for dynamic microsimulation models. Mimeo, Department of Work and Pensions, London.

Galler H., 1997. Discrete-time and Continuous-time Approaches to Dynamic Microsimulation Reconsidered'- NATSEM Technical Paper No. 13, -October. Canberra: National Centre for Social and Economic Modelling, University of Canberra, Australia.

Holm E, Lindgren U, Makila K, Malmberg G (1996), Simulating an entire nation, in G.P. Clarke (ed.), *Microsimulation for Urban and Regional Policy Analysis*, Pion, London, 164-186

Immervoll, H. and C. O'Donoghue, 2001, 'Towards a Multi-Purpose Framework for Tax-Benefit Microsimulation: A Discussion by Reference to EUROMOD, a European Tax-Benefit Model', *EUROMOD Working Paper*, EM2/01, Department of Applied Economics, University of Cambridge.

Leombruni, R. and M. Richiardi, 2005. An agent-based microsimulation of labour force participation. Some results for Italy, *mimeo, LABORatorio R. Revelli, University of Turin*.

Nolan B. 1990, “Inequity in the Financing and Delivery of Health Care in Ireland”, *ESRI Working Paper 16*, Economic and Social Research Institute, Dublin.

O’Donoghue C. 2001, “Dynamic Microsimulation: A Survey”, *Brazilian Electronic Journal of Economics*.

O’Donoghue C., 2002, “Redistribution over the Lifetime in the Irish Tax-Benefit System: An Application of a Prototype Dynamic Microsimulation Model for Ireland”, *Economic and Social Review*, Vol. 32, No. 3.

O’Donoghue C., D. Ballas, G. Clarke and J. Lennon, 2005. “Location Choice Decisions in Ireland”, paper presented to the conference *Modelling Urban Social Demographics*, University of Surrey, Guildford, March.

O’Donoghue C., 2005. “Assessing the Impact of Pensions Policy Reform in Ireland: the Case of Increasing the Pension Age” in E. Fornero and P. Sestito (eds.) *Pension Systems: Beyond Mandatory Retirement*. Cheltenham: Edward Elgar.

Orcutt G., J. Merz and H. Quinke, 1986. *Microanalytic Simulation Models to Support Social and Financial Policy*, Amsterdam: North-Holland.

PRIM 1997. *Assessing Policies for Retirement Income: Needs for Data, Research, and Models*. Washington DC: National Research Council.

Pudney S.E., 1992. “Dynamic Simulation of Pensioner’s incomes: methodological issues and a model design for Great Britain.”, Dept. of Applied Economics Discussion paper no MSPMU 9201, University of Cambridge.

Sauerbier Thomas, 2002, UMDBS - A New Tool for Dynamic Microsimulation. Journal of Artificial Societies and Social Simulation vol. 5, no. 2.

Wolfson M. and G. Rowe, 1998. “LifePaths – Toward an Integrated Microanalytic Framework for Socio-Economic Statistics”, paper presented to the *26th General Conference of the International Association for Research in Income and Wealth*, Cambridge, UK.

Appendix 1.

In LIAM, macro alignment occurs by specifying:

1. Alignment Sheets need to be the same shape for each process (but macro can be a subset), as does the predictor

2. Create Temporary Set of Alignment Structures of type talign (= n+1, where n is the number of processes to be macro aligned - structure (0) is to store the macro level)

3. For each sub process, run through conditions and count the number of people (level.nPer) who meet conditions who are in each alignment cell (we don't store predicted probability at this point as we don't know it - maybe simply assign zero and use the existing code)

4. For macro process, do the same

5. Multiply the cell p times the number in cell N = np, the number to be selected in cell

6. If the sub-processes are more disaggregated than the macro level, collapse to the lower level by summing N over the higher level (ie across education levels)

7. Now we have the N's for the 2 dimensional table for macro and each sub-process. Sum over sub-processes to get expected overall N_t and compare with the Macro N_m

8. To adjust multiply the highest level of the macro sheet (in this case level 2) In each of the sub-process by N_m/N_t

9. Backup original Alignment numbers (to be used in the following year)

10. Store new Alignment totals in the sub-process alignment structures.

Figures

Figure 1. Description of a Dynamic Module

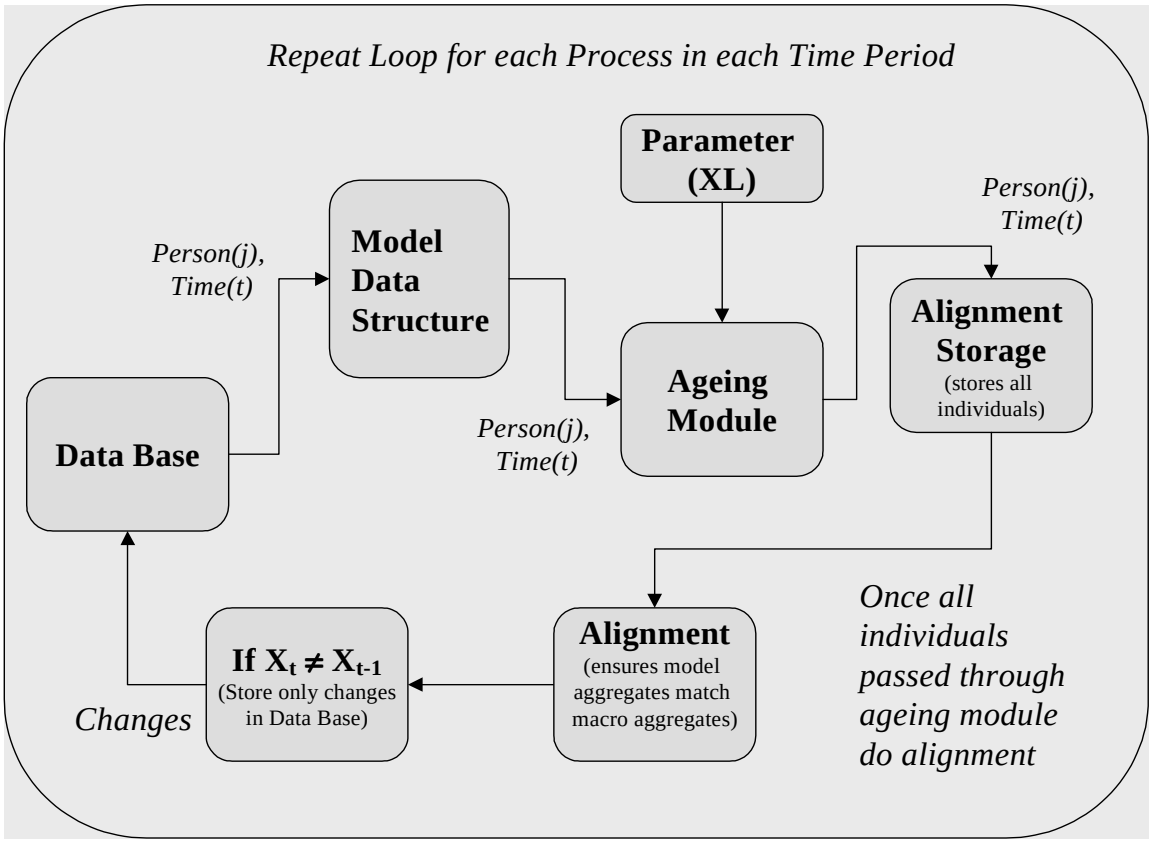
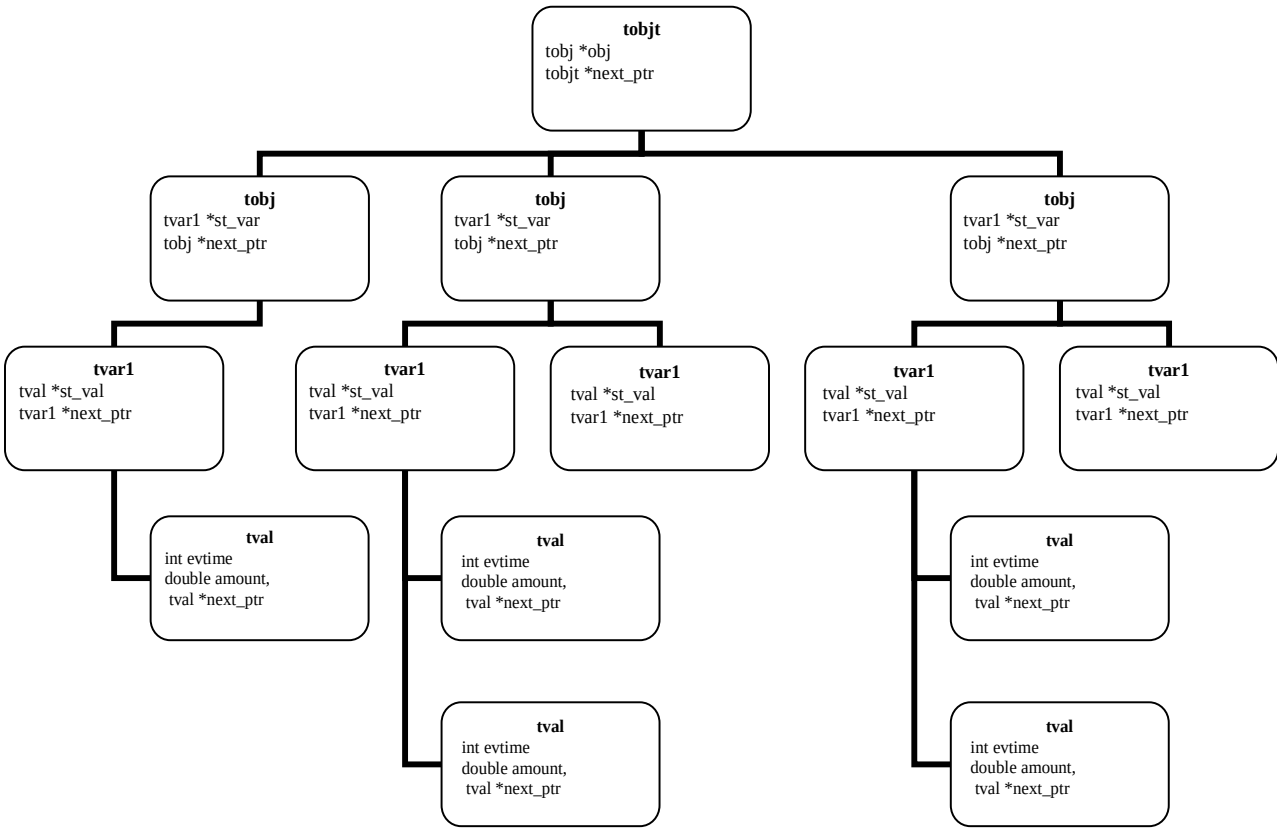
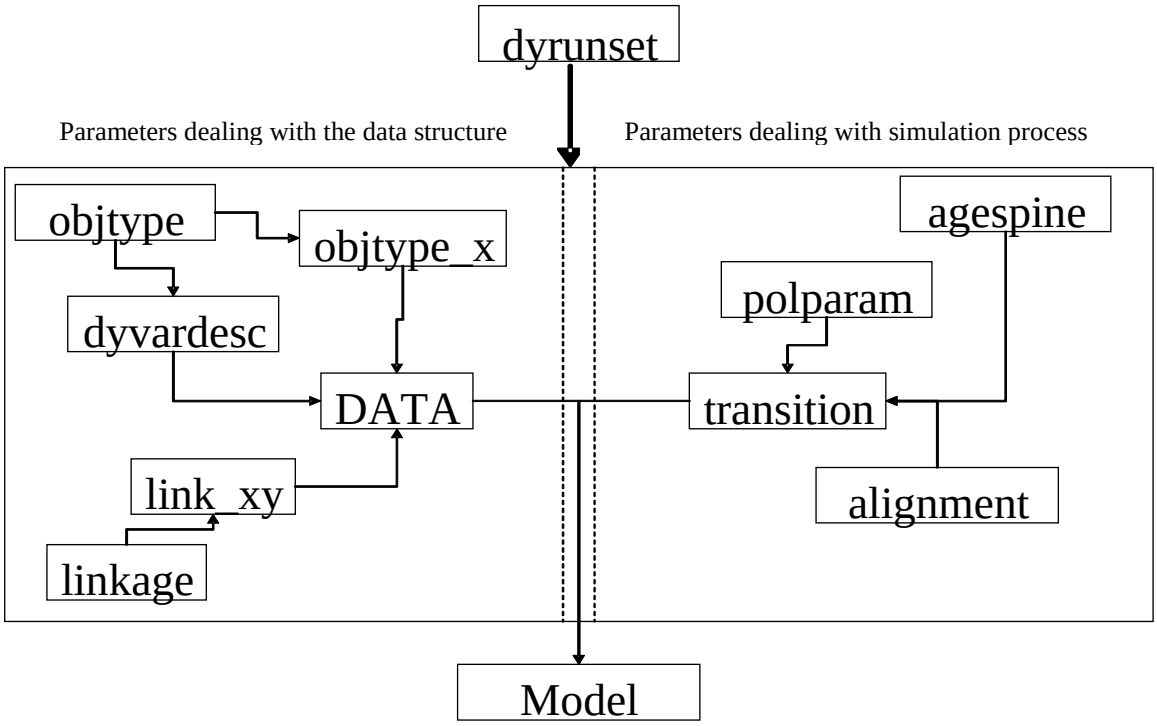


Figure 2. Model Data Structure



- The data is stored in a hierarchy of object types **(tobjt)** such as person, household, firm etc.
- Each of these object types themselves consists of a number of objects **(tobj)** such as the actual incidence of a person or household.
- Events **(tvar1)** such as births, tenure status or identification number then occur to objects.
- Each event can have a number of incidences or values **(tval).**
- For each event it is necessary to know what event occurred **(tvar1)**, when it occurred **(tval.evtime)**, and the value of the event **(tval.amount)**

Figure 3. Parameter Sheet Hierarchy



***dyrunset*:** parameters necessary to run the model, input/output directories, time period to be run etc.

***objtype*:** This file tells the model how many object types there are (region, household, person etc.).

***objtype_x*:** contains the incidences of each of the object types (r, h, p, etc.). So objtype_p would contain the set of id's of all persons.

dyvardesc: In this file, all variables used in the model framework are declared and described.

linkage: these parameters define the set of possible links between objects.

***link_xy*:** Stores the linkages between objects.

agespine: contains the list of modules to be run in the dynamic model.

transition: Each module has a corresponding parameter sheet in the parameter file "Transitions". These parameter files tell the model the output variables of each process, what type of process, whether a process needs to be aligned and the actual process parameters themselves such as the transition rates, regression equation and policy rules etc.

***alignment*:** If a particular process is to be aligned, then the model framework will look for an appropriate set of *alignment* parameters.

***polparam*:** this parameter set contains information associated with each parameter for each possible "system", where the system to be run is defined in the *dyrunset* parameters.

See Parameterisation section for more detailed description.

Our papers can be downloaded at:

http://cerp.unito.it/publications

CeRP Working Paper Series

N° 1/00	Guido Menzio	Opting Out of Social Security over the Life Cycle
N° 2/00	Pier Marco Ferraresi Elsa Fornero	Social Security Transition in Italy: Costs, Distorsions and (some) Possible Correction
N° 3/00	Emanuele Baldacci Luca Inglese	Le caratteristiche socio economiche dei pensionati in Italia. Analisi della distribuzione dei redditi da pensione (only available in the Italian version)
N° 4/01	Peter Diamond	Towards an Optimal Social Security Design
N° 5/01	Vincenzo Andrietti	Occupational Pensions and Interfirm Job Mobility in the European Union. Evidence from the ECHP Survey
N° 6/01	Flavia Coda Moscarola	The Effects of Immigration Inflows on the Sustainability of the Italian Welfare State
N° 7/01	Margherita Borella	The Error Structure of Earnings: an Analysis on Italian Longitudinal Data
N° 8/01	Margherita Borella	Social Security Systems and the Distribution of Income: an Application to the Italian Case
N° 9/01	Hans Blommestein	Ageing, Pension Reform, and Financial Market Implications in the OECD Area
N° 10/01	Vincenzo Andrietti and Vincent Hildebrand	Pension Portability and Labour Mobility in the United States. New Evidence from the SIPP Data
N° 11/01	Mara Faccio and Ameziane Lasfer	Institutional Shareholders and Corporate Governance: The Case of UK Pension Funds
N° 12/01	Roberta Romano	Less is More: Making Shareholder Activism a Valuable Mechanism of Corporate Governance
N° 13/01	Michela Scatigna	Institutional Investors, Corporate Governance and Pension Funds
N° 14/01	Thomas H. Noe	Investor Activism and Financial Market Structure
N° 15/01	Estelle James	How Can China Solve ist Old Age Security Problem? The Interaction Between Pension, SOE and Financial Market Reform
N° 16/01	Estelle James and Xue Song	Annuities Markets Around the World: Money's Worth and Risk Intermediation
N° 17/02	Richard Disney and Sarah Smith	The Labour Supply Effect of the Abolition of the Earnings Rule for Older Workers in the United Kingdom
N° 18/02	Francesco Daveri	Labor Taxes and Unemployment: a Survey of the Aggregate Evidence
N° 19/02	Paolo Battocchio Francesco Menoncin	Optimal Portfolio Strategies with Stochastic Wage Income and Inflation: The Case of a Defined Contribution Pension Plan
N° 20/02	Mauro Mastrogiacomo	Dual Retirement in Italy and Expectations
N° 21/02	Olivia S. Mitchell David McCarthy	Annuities for an Ageing World

N° 22/02	Chris Soares Mark Warshawsky	Annuity Risk: Volatility and Inflation Exposure in Payments from Immediate Life Annuities
N° 23/02	Ermanno Pitacco	Longevity Risk in Living Benefits
N° 24/02	Laura Ballotta Steven Haberman	Valuation of Guaranteed Annuity Conversion Options
N° 25/02	Edmund Cannon Ian Tonks	The Behaviour of UK Annuity Prices from 1972 to the Present
N° 26/02	E. Philip Davis	Issues in the Regulation of Annuities Markets
N° 27/02	Reinhold Schnabel	Annuities in Germany before and after the Pension Reform of 2001
N° 28/02	Luca Spataro	New Tools in Micromodeling Retirement Decisions: Overview and Applications to the Italian Case
N° 29/02	Marco Taboga	The Realized Equity Premium has been Higher than Expected: Further Evidence
N° 30/03	Bas Arts Elena Vigna	A Switch Criterion for Defined Contribution Pension Schemes
N° 31/03	Giacomo Ponzetto	Risk Aversion and the Utility of Annuities
N° 32/04	Angelo Marano Paolo Sestito	Older Workers and Pensioners: the Challenge of Ageing on the Italian Public Pension System and Labour Market
N° 33/04	Elsa Fornero Carolina Fugazza Giacomo Ponzetto	A Comparative Analysis of the Costs of Italian Individual Pension Plans
N° 34/04	Chourouk Houssi	Le Vieillissement Démographique : Problématique des Régimes de Pension en Tunisie
N° 35/04	Monika Bütler Olivia Huguenin Federica Teppa	What Triggers Early Retirement. Results from Swiss Pension Funds
N° 36/04	Laurence J. Kotlikoff	Pensions Systems and the Intergenerational Distribution of Resources
N° 37/04	Jay Ginn	Actuarial Fairness or Social Justice? A Gender Perspective on Redistribution in Pension Systems
N° 38/05	Carolina Fugazza Federica Teppa	An Empirical Assessment of the Italian Severance Payment (TFR)
N° 39/05	Anna Rita Bacinello	Modelling the Surrender Conditions in Equity-Linked Life Insurance
N° 40/05	Carolina Fugazza Massimo Guidolin Giovanna Nicodano	Investing for the Long-Run in European Real Estate. Does Predictability Matter?
N° 41/05	Massimo Guidolin Giovanna Nicodano	Small Caps in International Equity Portfolios: The Effects of Variance Risk.
N° 42/05	Margherita Borella Flavia Coda Moscarola	Distributive Properties of Pensions Systems: a Simulation of the Italian Transition from Defined Benefit to Defined Contribution
N° 43/05	John Beshears James J. Choi David Laibson Brigitte C. Madrian	The Importance of Default Options for Retirement Saving Outcomes: Evidence from the United States

N° 44/05	Henrik Cronqvist	Advertising and Portfolio Choice
N° 45/05	Claudio Campanale	Increasing Returns to Savings and Wealth Inequality
N° 46/05	Annamaria Lusardi Olivia S. Mitchell	Financial Literacy and Planning: Implications for Retirement Wellbeing
N° 47/06	Michele Belloni Carlo Maccheroni	Actuarial Neutrality when Longevity Increases: An Application to the Italian Pension System
N° 48/06	Onorato Castellino Elsa Fornero	Public Policy and the Transition to Private Pension Provision in the United States and Europe
N° 49/06	Mariacristina Rossi	Examining the Interaction between Saving and Contributions to Personal Pension Plans. Evidence from the BHPS
N° 50/06	Andrea Buffa Chiara Monticone	Do European Pension Reforms Improve the Adequacy of Saving?
N° 51/06	Giovanni Mastrobuoni	The Social Security Earnings Test Removal. Money Saved or Money Spent by the Trust Fund?
N° 52/06	Luigi Guiso Tullio Jappelli	Information Acquisition and Portfolio Performance
N° 53/06	Giovanni Mastrobuoni	Labor Supply Effects of the Recent Social Security Benefit Cuts: Empirical Estimates Using Cohort Discontinuities
N° 54/06	Annamaria Lusardi Olivia S. Mitchell	Baby Boomer Retirement Security: The Roles of Planning, Financial Literacy, and Housing Wealth
N° 55/06	Antonio Abatemarco	On the Measurement of Intra-Generational Lifetime Redistribution in Pension Systems
N° 56/07	John A. Turner Satyendra Verma	Why Some Workers Don't Take 401(k) Plan Offers: Inertia versus Economics
N° 57/07	Giovanni Mastrobuoni Matthew Weinberg	Heterogeneity in Intra-Monthly Consumption. Patterns, Self-Control, and Savings at Retirement
N° 58/07	Elisa Luciano Jaap Spreeuw Elena Vigna	Modelling Stochastic Mortality for Dependent Lives
N° 59/07	Riccardo Calcagno Roman Kraeussl Chiara Monticone	An Analysis of the Effects of the Severance Pay Reform on Credit to Italian SMEs
N° 60/07	Riccardo Cesari Giuseppe Grande Fabio Panetta	La Previdenza Complementare in Italia: Caratteristiche, Sviluppo e Opportunità per i Lavoratori
N° 61/07	Irina Kovrova	Effects of the Introduction of a Funded Pillar on the Russian Household Savings: Evidence from the 2002 Pension Reform
N° 62/07	Margherita Borella Elsa Fornero Mariacristina Rossi	Does Consumption Respond to Predicted Increases in Cash-on-hand Availability? Evidence from the Italian "Severance Pay"
N° 63/07	Claudio Campanale	Life-Cycle Portfolio Choice: The Role of Heterogeneous Under-Diversification
N° 64/07	Carlo Casarosa Luca Spataro	Rate of Growth of Population, Saving and Wealth in the Basic Life-cycle Model when the Household is the Decision Unit
N° 65/07	Annamaria Lusardi	Household Saving Behavior: The Role of Literacy, Information and Financial Education Programs (Updated version June 08: "Financial Literacy: An Essential Tool for Informed Consumer Choice?")

N° 66/07	Maarten van Rooij Annamaria Lusardi Rob Alessie	Financial Literacy and Stock Market Participation
N° 67/07	Carolina Fugazza Maela Giofré Giovanna Nicodano	International Diversification and Labor Income Risk
N° 68/07	Massimo Guidolin Giovanna Nicodano	Small Caps in International Diversified Portfolios
N° 69/07	Carolina Fugazza Massimo Guidolin Giovanna Nicodano	Investing in Mixed Asset Portfolios: the Ex-Post Performance
N° 70/07	Radha Iyengar Giovanni Mastrobuoni	The Political Economy of the Disability Insurance. Theory and Evidence of Gubernatorial Learning from Social Security Administration Monitoring
N° 71/07	Flavia Coda Moscarola	Women participation and caring decisions: do different institutional frameworks matter? A comparison between Italy and The Netherlands
N° 72/08	Annamaria Lusardi Olivia Mitchell	Planning and Financial Literacy: How Do Women Fare?
N° 73/08	Michele Belloni Rob Alessie	The Importance of Financial Incentives on Retirement Choices: New Evidence for Italy
N° 74/08	Maela Giofré	Information Asymmetries and Foreign Equity Portfolios: Households versus Financial Investors
N° 75/08	Harold Alderman Johannes Hoogeveen Mariacristina Rossi	Preschool Nutrition and Subsequent Schooling Attainment: Longitudinal Evidence from Tanzania
N° 76/08	Riccardo Calcagno Elsa Fornero Mariacristina Rossi	The Effect of House Prices on Household Saving: The Case of Italy
N° 77/08	Giovanni Guazzarotti Pietro Tommasino	The Annuity Market in an Evolving Pension System: Lessons from Italy
N° 78/08	Margherita Borella Giovanna Segre	Le pensioni dei lavoratori parasubordinati: prospettive dopo un decennio di gestione separata
N° 79/08	Annamaria Lusardi	Increasing the Effectiveness of Financial Education in the Workplace
N° 80/08	Claudio Campanale	Learning, Ambiguity and Life-Cycle Portfolio Allocation
N° 81/09	Fabio Bagliano Claudio Morana	Permanent and Transitory Dynamics in House Prices and Consumption: Cross-Country Evidence
N° 82/09	Carolina Fugazza Massimo Guidolin Giovanna Nicodano	Time and Risk Diversification in Real Estate Investments: Assessing the Ex Post Economic Value
N° 83/09	Annamaria Lusardi Peter Tufano	Debt Literacy, Financial Experiences, and Overindebtedness
N° 84/09	Luca Spataro	Il sistema previdenziale italiano dallo shock petrolifero del 1973 al Trattato di Maastricht del 1993
N° 85/09	Cathal O'Donoghue John Lennon Stephen Hynes	The Life-Cycle Income Analysis Model (LIAM): A Study of a Flexible Dynamic Microsimulation Modelling Computing Framework