Dypes
Dynamic microsimulation of pensions

- **The Dypes model: Overview and Features**
- **The microsimulation programing language Modgen: Overview and Features**
- **User Interfaces**
- **Key language concepts**

Dypes overview

- **Introduction**

The majority of developed countries are facing a population aging process that may threat the sustainability of their social protection programmes. In a context of rising long-term care costs and increasing pension's bill, the concern about the necessary reforms to make the system sustainable is fully justified. Reform proposals vary from a complete restructuring of the system – like a switch to a true or to a notional capitalisation system – to marginal adjustments of the legal parameters of the current system.

The dynamic microsimulation model Dypes is a useful tool to analyse the effects of pension system reforms. The flexibility of the model allows to focus in different aspects of the reforms:

- Consider both the system sustainability and pension level adequacy,
- Looking at the differential impact of changes in financial incentives by population groups (males and females and different working careers).

The first issues conform the two main pillars of the European Commission concerns in terms of pension's policies. The second can be useful to design appropriate responses from the private sector (mainly retirees).

- **Description**

Dypes simulates in continuous time the main events of individual's life up to 2060: education, labour entry, labour market transitions, unemployment spells and, finally, retirement and mortality. It employs available information on the probabilities of each transition by population groups and –unlike other microsimulation models- it uses a

starting population sample with information on working careers to improve simulation results. Dypes starts from an administrative sample of individuals related to Social Security (Muestra Continua de Vidas Laborales: http://www.seg-social.es/Internet_1/Estadistica/Est/Muestra_Continua_de_Vidas_Laborales/index.htm). Dypes also incorporates advanced behavioural modules to capture individual's reactions to both business cycle fluctuations and changes in retirement incentives.

- **Advantages**

- Flexibility given the modular structure.
- The model accounts for individuals' reactions to changes in financial incentives to retire. It is therefore a tool particularly suitable for analysing reform effects focusing in different population groups.
- The main events are modelled in continuous time.
- Dypes uses a starting population sample to better feet with the reality.
-

- **Current stage of development**

Dypes is currently working both in time based and case based version. It has been used to analyze the effects of the 2011 and 2013 reforms of the Spanish pension system.

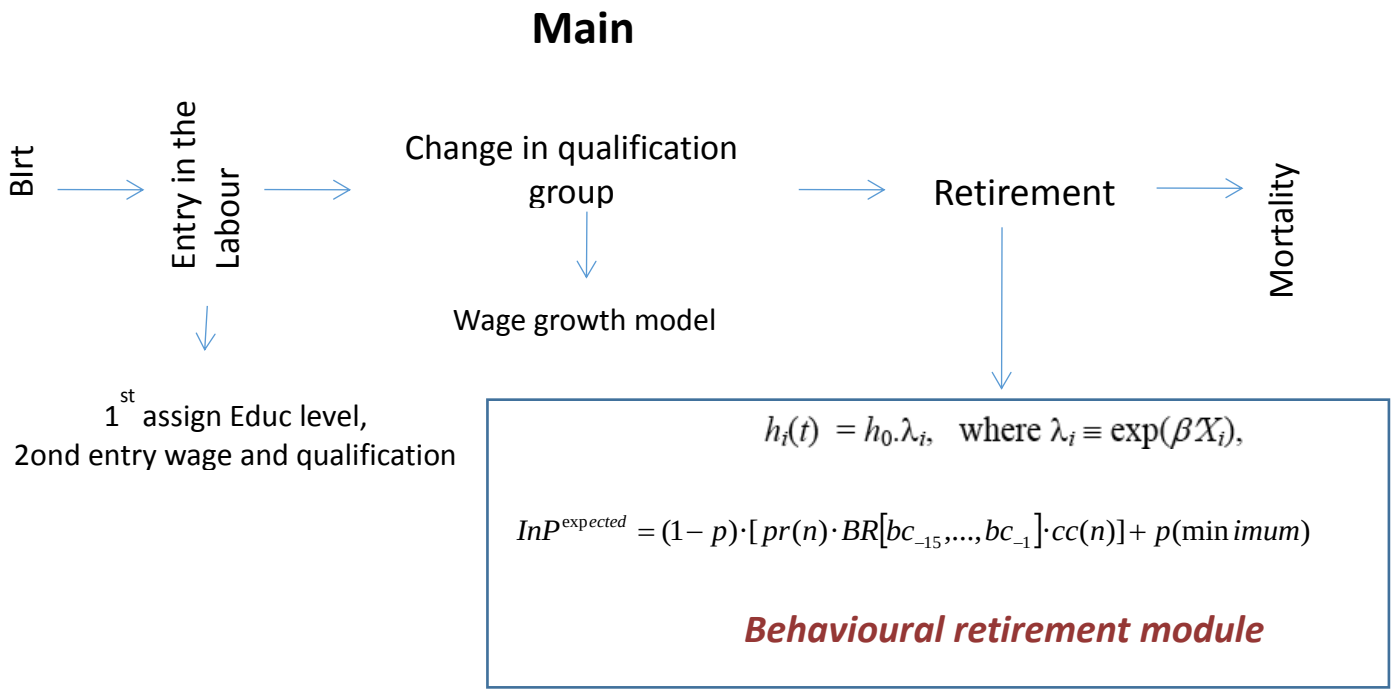
Dypes: Features

Dynamic population microsimulation model

1. Runs on continuous time - some events occur in discrete time.
2. Starting population subsample. Based on an **administrative data set**: 1,200,000 individuals (not households) from Social Security registers (the 2007 wave of the MCVL -*Muestra continua de vidas laborales*). Future generations are simulated (1,800,000 total individuals).
3. Both versions: case-based and time-based model: The first version (Fernández-Díaz, Patxot, Souto, 2013)[1] is "case-based". It simulates each case from birth to death before the simulation of the next case begins (case-based model)-, and the second version (Patsot, Solé, Souto, 2017)[2] is "time-based" and hence simulates every year all cases taking into account more explicitly interactions.
4. Behavioral approach for some key events of the model: retirement, based on expected pension given legislation and unemployment expectations.

[1] Fernández-Díaz, F. J., Patxot, C., & Souto, G. (2013). DYPES: A Microsimulation model for the Spanish retirement pension system. *Documento de Trabajo*, 06.
[2] Patxot, C., Solé, M., & Souto, G. (2017). Should pensions be redistributive? The impact of Spanish reforms on the system's sustainability and adequacy. *Documento de Trabajo*, 02. **http://documentos.fedea.net/pubs/dt/2017/dt2017-02.pdf**

Dypes structure



Trade-off between increasing behavioural realism and keep the model treatable

Modgen overview

- Modgen is a programming language developed in Statistics Canada and used in dozens of applications around the globe

- Modgen is freely available from Statistics Canada

- Currently, there are two versions of Modgen: one needs Visual Studio 2010 but an open-source implementation of Modgen openM++ have been recently available (https://ompp.sourceforge.io/wiki/index.php/Main_Page)

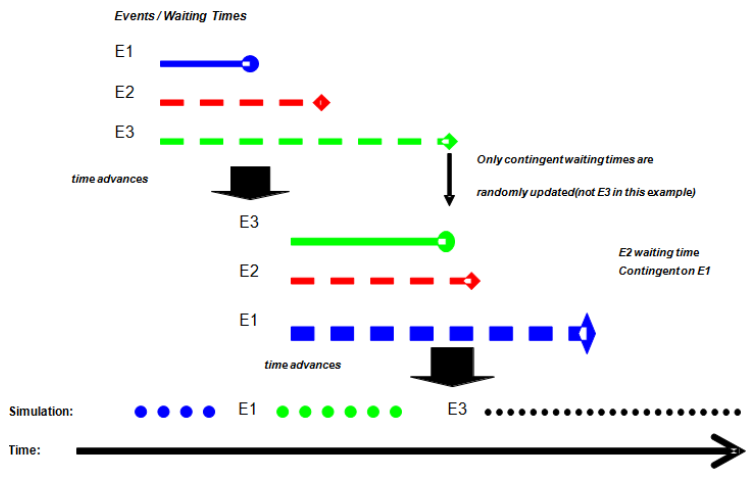
- Modgen creates a stand-alone model executable program with a complete visual interface and detailed model documentation

- Modgen translates Modgen code into C++. This allows to combine the strengths of the generic C++ language with specialized microsimulation language concepts and functions
- Minimizes code writing as the general underlying mechanisms in dynamic microsimulation are already programed and hidden

- Modgen includes a powerful tabling language and tools for visualizations.

Modgen: Features

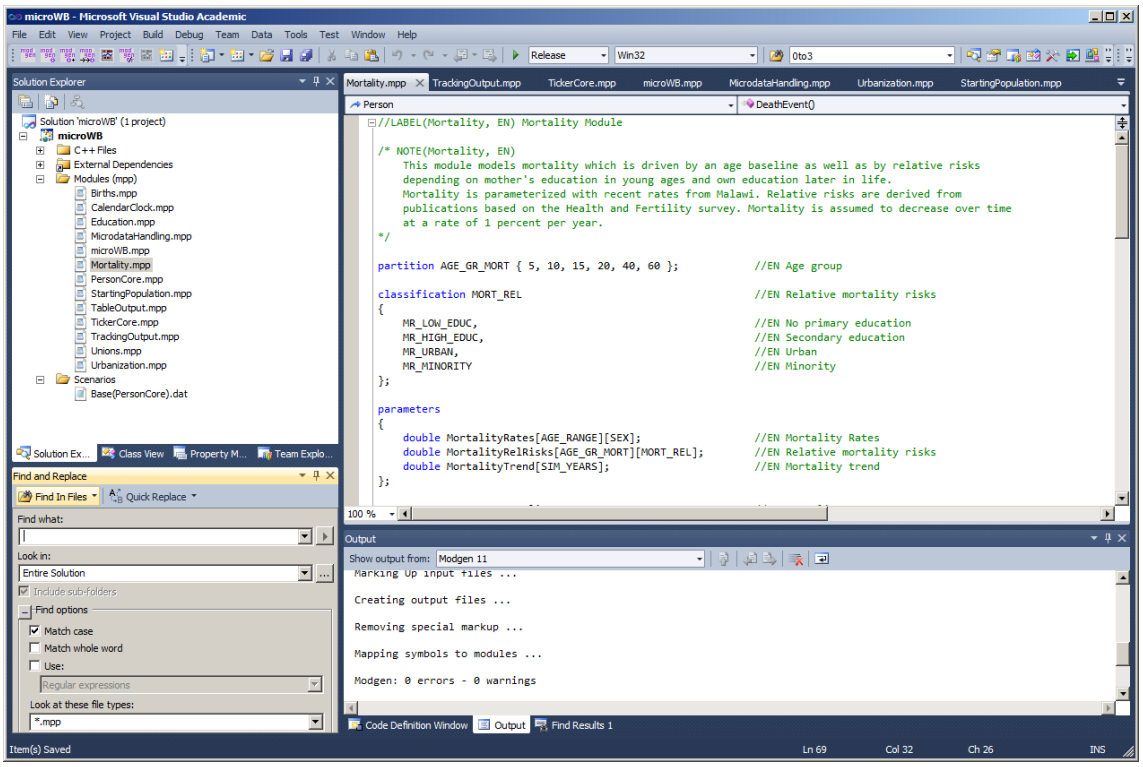
- It allows all approaches in dynamic microsimulation: Discrete, or **continuous** time; interacting or non-interacting populations; case-based or time-based
- It allows using a starting population subsample (if needed)
- Fast: compiled language, pre-compiled to C++. 35 mi 1,4 individuals, whole life.
- Multilingual models possible
- Export of parameters and tables to Excel
- Unlimited dimensions for parameters and tables
- Visualization of individual life courses
- Common fully documented user interface
- Scenario management
- Automated generation of model documentation
- Multi-threading and grid-computing possible
- ModGen way of working:
    - It creates an event queue
    - Computes time to event (using behavioural or fixed probabilities)
    - Once and event happens, all times (queue) revaluated



Why using ModGen in Dypes?

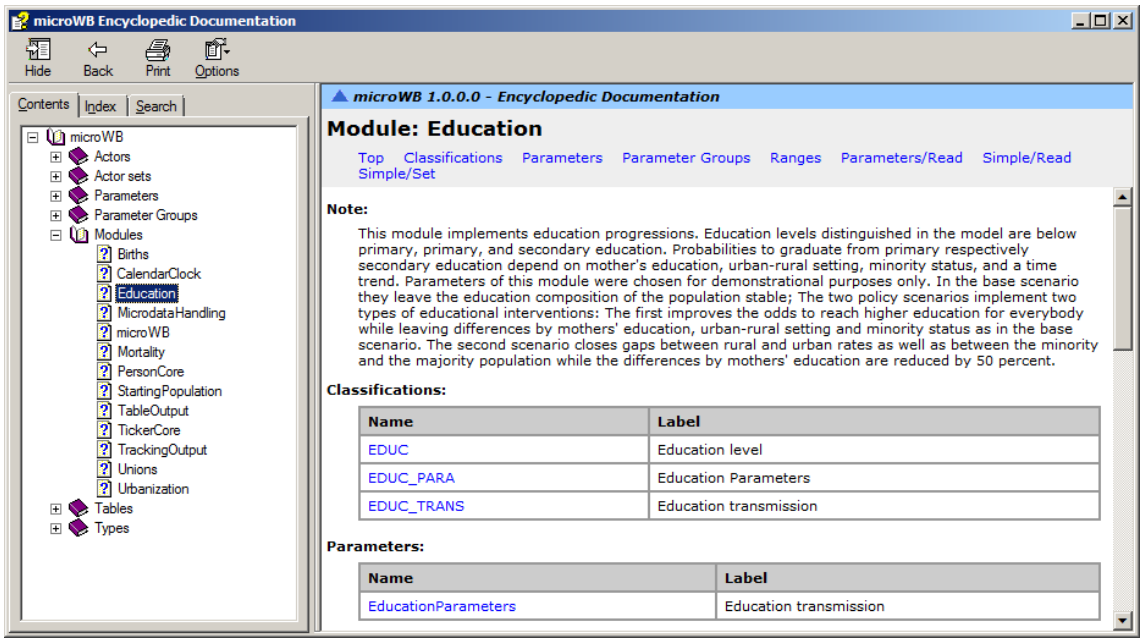
- Efficient and freely available programming language + user friendly application
  - Used in many different kind of models
  - **Relevant in Spain to handle huge data bases like the MCV (1,8 million people takes 30 mi)**
  - Based in C++ and implemented in VisualStudio (soon open source openM+, end 2015)
  - Minimizing code writing keeping flexibility → It frees up for other tasks: Data mining, behavioural analysis
- Easy conversion from case based (longitudinal) to time based (cross-section)
  - Advantages of longitudinal: Lower computational requirements and allows for simulating the effects of the 2013 reform of the Spanish pension system.
  - Advantages of cross-section: Allows for interaction and hence, it eases:
    - Incorporates computation of relative measures like inequality indicators (Ginis, etc)
    - Alignments

Programming interface



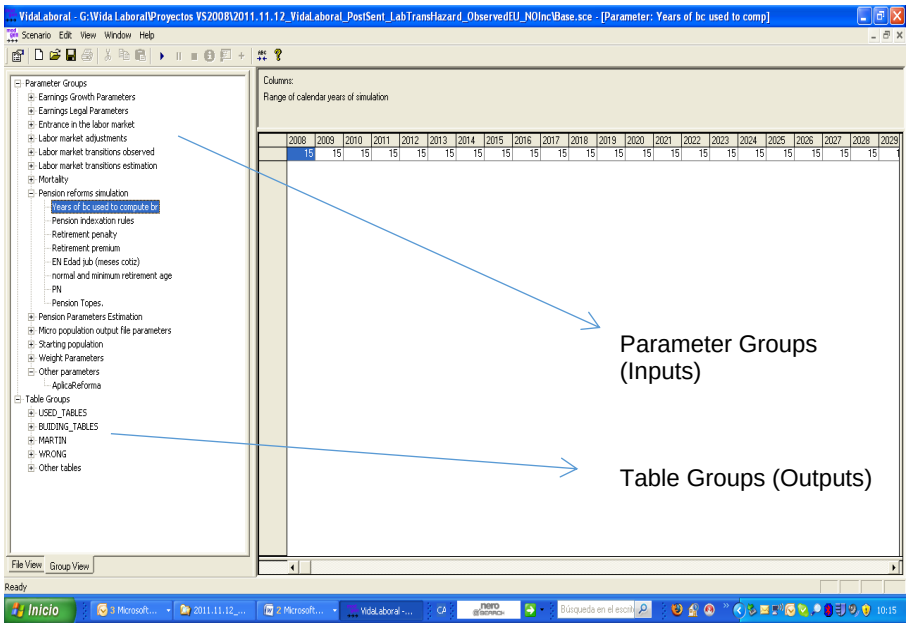
The main window shows the code of a particular file, while the list of files (modules) created is in the left window.

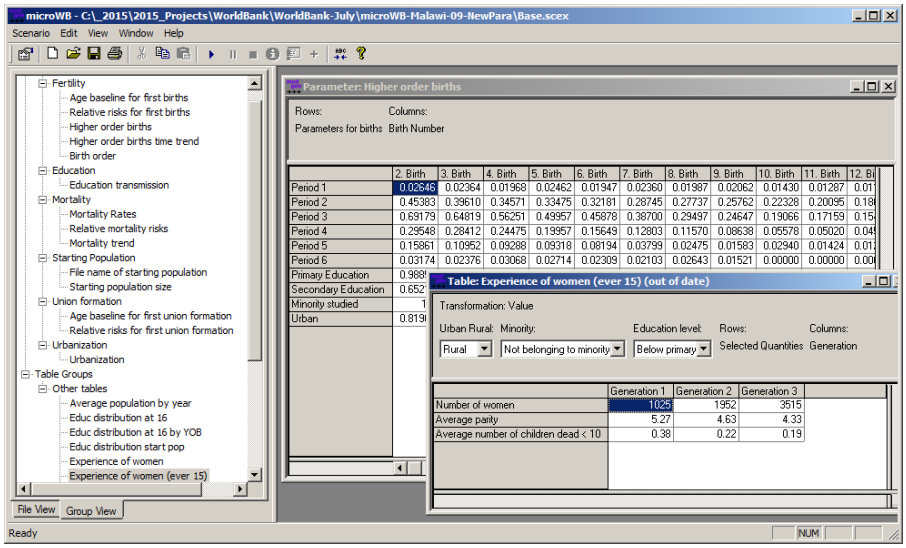
Model documentation



The executable application includes and automatically created documentation. As an example the image shows the content of a module called “Education” showing in the main window, a brief description and the elements (classifications, parameters, etc.) involved in the module. Those can also be directly approached in the left window.

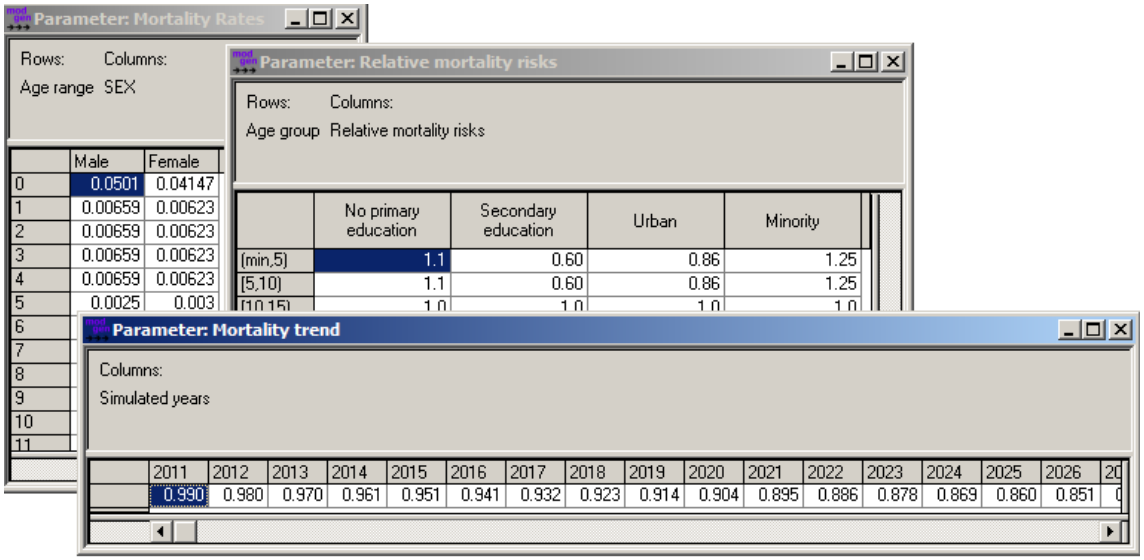
User interface of the executable application



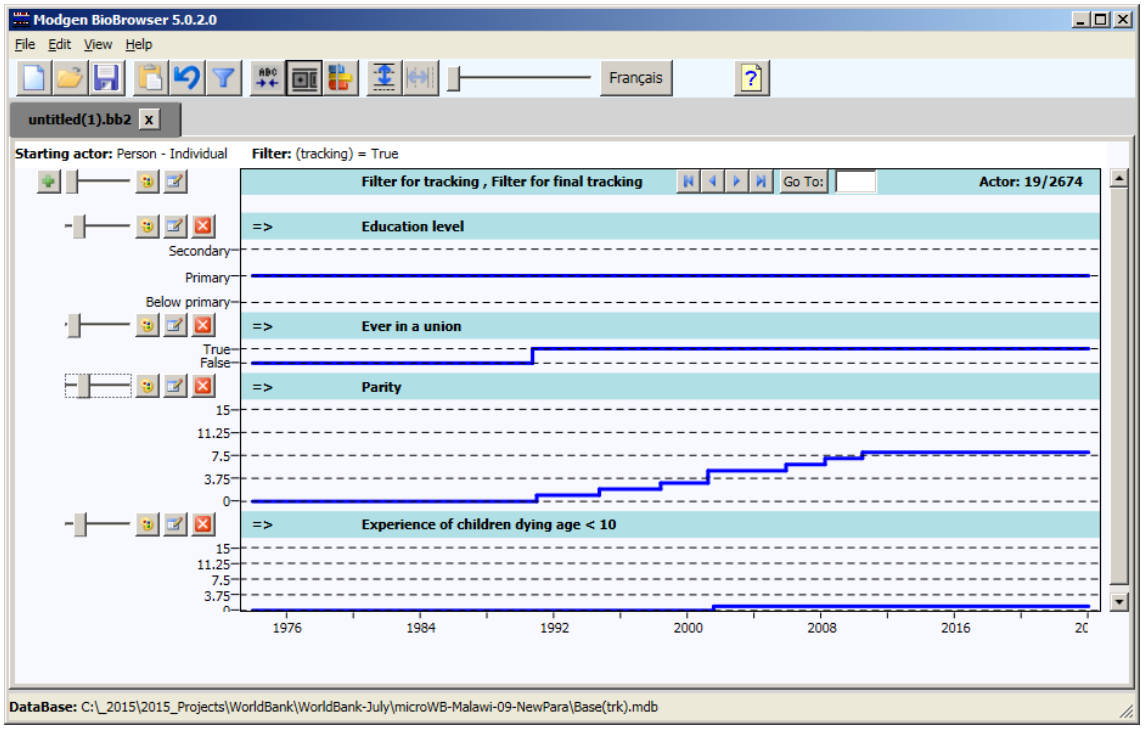


The user interface allows to access the inputs and outputs of the model, which appear as tables in the left window. The input tables can be changed directly in order to run alternative scenarios.

Resulting parameter tables



BioBrowser



There Biobrowser allows to observe the transitions of concrete individuals. For example, education level remains constant, while other variables experimenting changes during the life course show changing values.

Key ModGen concepts and programing

The following are the elements that need to be programed.

- Parameters: organized in tables (which can be of any dimension). Data are stored in .dat files and can be edited within the user interface.
- Actors: An actor is the entity whose life is simulated. A model's actor is often a person
- States: describe the characteristics of actors. States can be logical, numeric, or categorical.
- Events: consists of two functions:
    - Time function: when does event happen?
    - Implementation function: the consequences when the event happens.

- **Declaration of states and events**

As an example, the following code defines an actor and its age while it is alive.

```
actor Person
{
    logical alive    = { TRUE };                                    //EN Alive
    logical ever_15 = ( integer_age >= 15 );                        //EN Person 15+
    int     age_gr_mort = self_scheduling_split(age, AGE_GR_MORT); //EN Age  group
    event   timeDeathEvent, DeathEvent;                            //EN Death event
};
```

- **Types of states**

- Simple: Initialized, then updated in events
- Derived: derived from other states and automatically updated

- Self-scheduling: update themselves according a given schedule

As an example, the following code defines age groups and the parity status.

```
partition AGE_GROUP { 15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40 };

actor Person
{
    PARITY_STATUS   parity_status = { PS_CHILDLESS };                  //EN Parity

    integer         age_group = self_scheduling_split(age, AGE_GROUP);  //EN Age group

    //EN Pregnancy hazard
    double          pregnancy_hazard = ( parity_status == PS_CHILDLESS ) ?
                            FirstPregnancyBaseline[age_group]
                            * FirstPregnancyRelative[union_status] : 0.0;
```

- **Dimensions**

- Ranges: a limited set of integer numbers.
- Partitions: splitting continuous time into intervals
- Classifications: categorical variables

```
range TAB_AGE { 0, 100 };       //EN Allowed age range

//EN Age partition
partition AGE_GROUP { 15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40 };

classification PARITY_STATUS    //EN Partity Status
{
    PS_CHILDLESS,               //EN Childless
    PS_PREGNANT                 //EN Pregnant
};

parameters
{
    double  FirstPregnancyBaseline[AGE_GROUP];      //EN First pregnancy baseline risk
```

Code: dimensions and parameters

```
range       AGE_RANGE { 0, 100 };                          //EN Age range
range       SIM_YEARS { 2011, 2150 };                      //EN Simulated years

partition   AGE_GR_MORT { 5, 10, 15, 20, 40, 60 };         //EN Age group

classification SEX
{
    MALE,                                                  //EN Male
    FEMALE                                                 //EN Female
};

classification MORT_REL                                    //EN Relative mortality risks
{
    MR_LOW_EDUC,                                           //EN No primary education
    MR_HIGH_EDUC,                                          //EN Secondary education
    MR_URBAN,                                              //EN Urban
    MR_MINORITY                                            //EN Minority
};

parameters
{
    double MortalityRates[AGE_RANGE][SEX];                 //EN Mortality Rates
    double MortalityRelRisks[AGE_GR_MORT][MORT_REL];       //EN Relative mortality risks
    double MortalityTrend[SIM_YEARS];                      //EN Mortality trend
};
```

Event example: Fertility

```
actor Person
{
    PARITY_STATUS   parity_status = { PS_CHILDLESS };                  //EN Parity

    //EN Pregnancy hazard
    double          pregnancy_hazard = ( parity_status == PS_CHILDLESS ) ?
                            FirstPregnancyBaseline[age_group]
                            * FirstPregnancyRelative[union_status] : 0.0;

    event           timePregnancyEvent, PregnancyEvent;                //EN Pregnancy Event
};

TIME Person::timePregnancyEvent()
{
    if ( pregnancy_hazard > 0.0 ) return WAIT(-log(RandUniform(2)) / pregnancy_hazard);
    else return TIME_INFINITE;
}

void Person::PregnancyEvent()
{
    parity_status = PS_PREGNANT;
}
```

Tables

In order to define output tables the following elements need to be specified. In the following an example is shown of how a table is programed and the resulting image in the executable application.

- Name
- Filter
- Dimensions
- Expressions

Table definitions

```
table Person ChildlessByAge                          //EN Childlessness by age
[ WITHIN(TAB_AGE15, int_age) ]
{
    tab_age15 *
    {
        //EN Proportion childless at beginning of age year decimals=4
        value_in(is_childless) / unit,

        //EN Proportion lived childless by year of age decimals=4
        duration(is_childless,TRUE) / duration()
    }
};
```

```
table Person DeadChildren10_ever15_3GEN     //EN Experience of women (ever 15) 3 GEN
[ trigger_transitions(alive,TRUE,FALSE) && sex == FEMALE && generation >= 1  && generation <= 3 && ever_15 ]
{
    urban_rural+ *
    minority+ *
    educ+ *
    {
        unit,                                   //EN Number of women
        value_in(parity)/unit,                  //EN Average parity decimals=2
        value_in(child_deaths10)/unit           //EN Average number of children dead < 10 decimals=2
    }
    * gen1to3
};
```

Resulting table

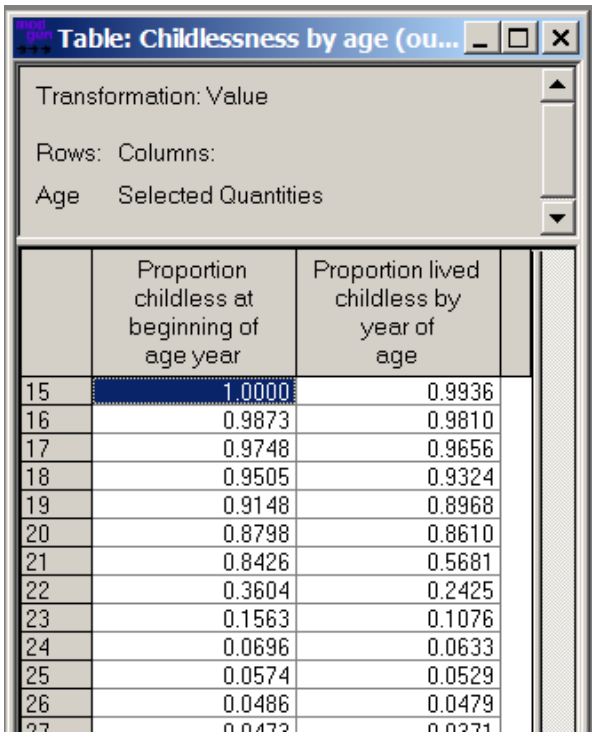


Table: Childlessness by age (ou...

Transformation: Value

Rows: Columns:

Age Selected Quantities

	Proportion childless at beginning of age year	Proportion lived childless by year of age
15	1.0000	0.9936
16	0.9873	0.9810
17	0.9748	0.9656
18	0.9505	0.9324
19	0.9148	0.8968
20	0.8798	0.8610
21	0.8426	0.5681
22	0.3604	0.2425
23	0.1563	0.1076
24	0.0696	0.0633
25	0.0574	0.0529
26	0.0486	0.0479

Information

Dypes documentation.

- **CHM File name:** NuevaVidaLaboralTB2_EN.chm

- Publications:

Fernández-Díaz, F. J., Patxot, C., & Souto, G. (2013). DYPES: A Microsimulation model for the Spanish retirement pension system. *Documento de Trabajo*, 06.
Patxot, C., Solé, M., & Souto, G. (2017). Should pensions be redistributive? The impact of Spanish reforms on the system's sustainability and adequacy. *Documento de Trabajo*, 02. **http://documentos.fedea.net/pubs/dt/2017/dt2017-02.pdf**

- **Starting sample:** http://www.seg-social.es/Internet_1/Estadistica/Est/Muestra_Continua_de_Vidas_Laborales/index.htm

Modgen downloads and documentation:

http://www.statcan.gc.ca/eng/microsimulation/modgen/modgen

OpenM++ implementation (open source):

https://ompp.sourceforge.io/wiki/index.php/Main_Page