System Models & Simulation

Harold Halbleib - Excel Software

Abstract

We live in a world of systems driven by cause and affect. Those systems include financial, production, inventory, biological, chemical, thermodynamic or workflow. Systems can be modelled as nodes representing system variables and connecting lines representing causal effects. The changing value of one variable can cause another to increase or decrease as described by equations.

Understanding how a system really works is the first step towards using, improving, automating or explaining it to others. This paper shows how to model dynamic systems, run time simulations and present the system behaviour with charts, graphs and tables. It provides example models for population growth, bank balance, demographics, production and water management.

Introduction

Causal Loop Diagrams (CLDs) are used to model dynamic systems. The simple diagram notation of nodes and lines identifies the important variables in a system and how they interact. The CLD presents an easily understood conceptual model of how the system works. Facts about the system are used to parameterize the model. Each node becomes a variable that identifies a quantifiable property of the system that changes over time. Each line can have equations or rules that formalize how one variable affects another.

A parameterized model has all the information needed to simulate the dynamic response of the system over a series of time increments. The model is declarative in that the diagram, variables and equations just declare facts about the system. Contrast that with a procedural approach written in a programming language like C where algorithms would have to be developed and implemented to simulate the dynamic system response.

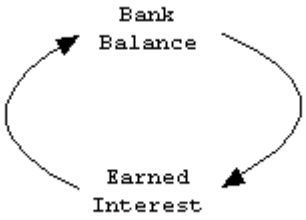
The WinA&D modelling tool provides a modelling environment that automatically generates a simulation script for the dynamic response to user controlled conditions and variable values. The output data can be viewed, exported or graphed with little human effort. Even if you never run a simulation, the mere process of creating the declarative model is very valuable. It forces you to think through and understand how the system really works.

Simulation is the process of starting with an initial value in each variable and running the set of equations for multiple time increments. At each time increment, all equations are processed to generate new variable values from the current values. The resulting data for variables at each time increment represents one run of the simulation.

During simulation, a model is driven by input data and produces output data. Data can come from initial values in the model, user entry or a data file. Output data can be stored in the model or in output data files and used to drive configured charts, graphs and tables. An integrated Data File editor makes it easy to create, view, import data from or export data to other applications.

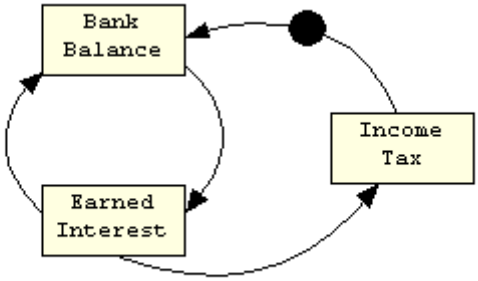
Causal Loop Diagram

A Causal Loop Diagram (CLD) identifies each variable in the system as a state or node and causal affects between nodes shown as lines. For example, your saving account could be shown as a simple system consisting of two nodes, Bank Balance and Earned Interest. The amount of the Bank Balance will determine the amount of the Earned Interest as represented by a solid line pointing from Bank Balance to Earned Interest.



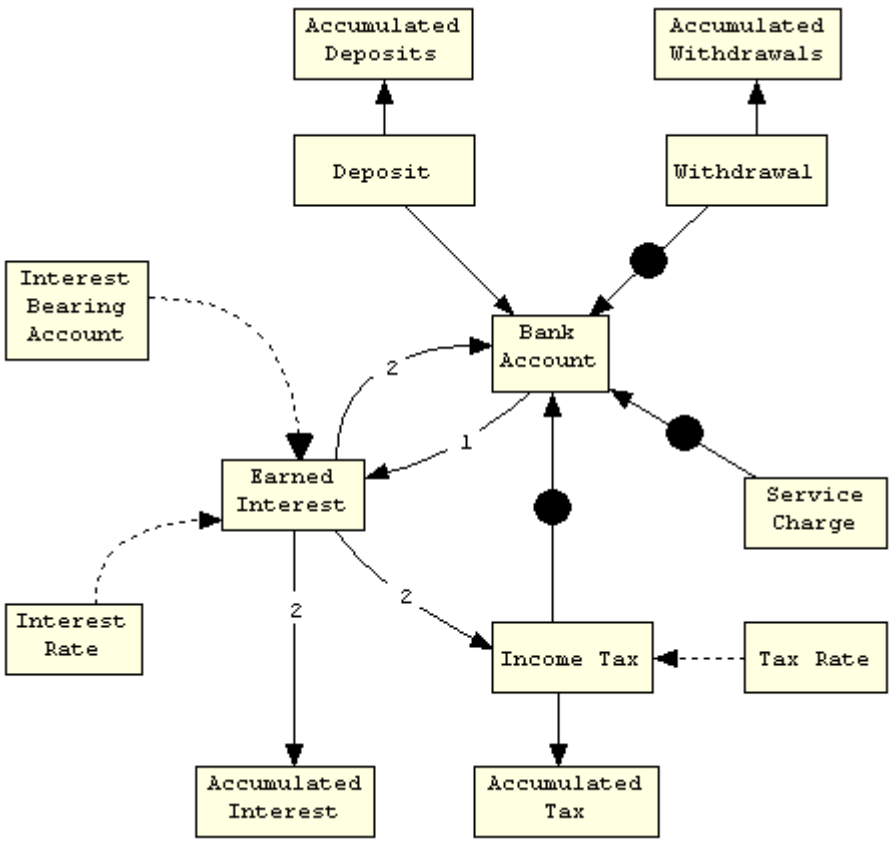
Positive Causal Affects

The causal affect between these nodes forms a positive reinforcing loop. As the Bank Balance grows, the Earned Interest value also grows at each time increment.



Negative Causal Affect from Taxation

As the value of Earned Interest grows, the Income Tax grows. That tax has a negative causal affect on the Bank Balance as indicated on the diagram with a solid dot notation on the line drawn from Income Tax to Bank Balance. Since the nodes in this model can be assigned objectively measured values, we'll use a solid box notation.



Causal Loop Diagram of Bank Account

The example above shows the causal affects of other variables involved in a bank account. Later we'll define equations for each solid line indicating how one variable affects another. Dashed lines don't have equations. They are controls that provide a variable for other equations on lines coming into that node. For example, the Interest Rate value affects how the Earned Interest value is calculated on the line coming from the Bank Account node.

As you can see, casual loop diagrams are simple to draw and understand. There are additional notational adornments offered by articles and books on system modelling, but they are just refinements on these simple concepts. WinA&D implements the most popular notations.

Parameterize Model

There are two main steps to parameterize a model, configuring the nodal variables and assign equations to the causal lines. In WinA&D, each node and line from the diagram creates a dictionary entry for the project. Configuration data is entered into a Details dialog to define the variables and equations.

For example, Bank Balance could be defined as a numeric variable with an initial value and units in dollars. Variables can also be configured as constants, calculated values, strings, arrays, list or Booleans. A variable can have a Minimum and Maximum range and be read from or written to data files during simulation.

The line from Bank Account to Earned Interest has a simple equation that determines how variable values change at each time increment.

EarnedInterest = InterestRate * BankAccount

In WinA&D, each node becomes a named variable. Equations use those names with integer, floating-point or boolean operators like "+", "/", "<", ">", etc. Equations can include arrays and standard math functions like sine, cosine, square root, exponentials or arrays for manipulating more complex data set.

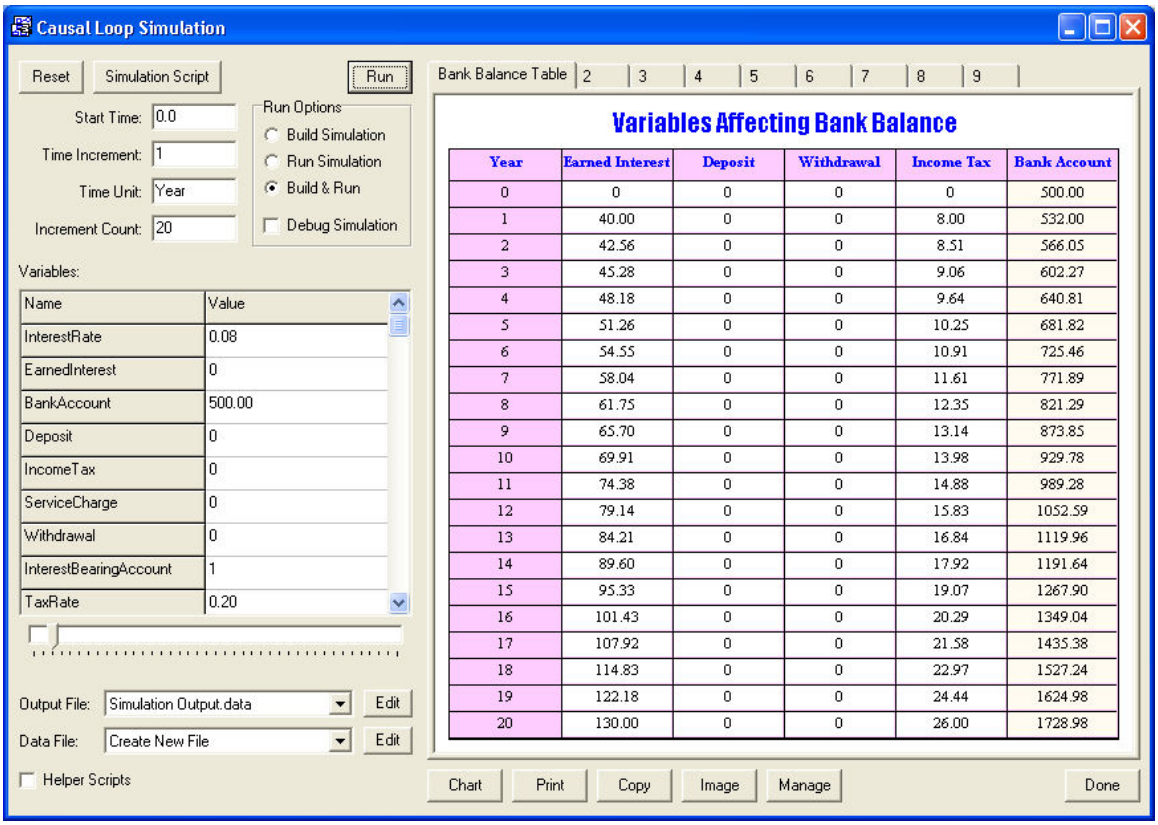
Each solid line in the model will have a list of one or more equations that define its causal affect. Sometimes, equations are order dependent. For example, Earned Interest must be calculated before Income Tax. That dependency can be shown on the diagram with a sequence number on the causal line.

After configuring variables and equations, a model is ready for simulation.

Simulate Model

WinA&D's Simulation command checks for completeness in the model, and then presents the Simulation dialog shown below. On the left side of the dialog, specify the Time Units and Increment Count, then set initial variable values as desired in the Variables table. Change variable values in the table or use the slider bar presented using the range of values previously configured.

Click the Run button to perform a simulation run using the assigned values. Results of the simulation are stored in a data file and can be presented in a configured table, chart or graph. In this example, important variable data is presented in a table. If you change variable values like interest rate, tax rate, etc and click the Run button again, you'll add another run of simulation data to the output file. A configured chart can show a specific data run, the last run or multiple runs at once.



Simulation of Bank Account with Results Presented in Configured Table

Scripted Simulation Rules

During simulation, WinA&D generates a script that represents the dynamic system then runs that script for a series of time increments. During script generation, each user entered list of equations is automatically converted into WinA&D's built-in scripting language.

In a sophisticated simulation, the user can directly define the piece of the script that implements a causal line in the model. For example, the equation:

EarnedInterest = InterestRate * BankAccount

is converted to the script:

%%SV"EarnedInterest = InterestRate * BankAccount"

In the Details dialog for a causal line, the user can switch back and forth between the Equation or Script definition.

WinA&D's built-in scripting language opens a world of possibilities when creating complex models and simulations. Scripts can perform calculations with variables, arrays or list, do string manipulations, do conditional logic, call routines, grab data from any WinA&D document, read and write data from files and virtually anything else that's possible from a typical programming language. The scripting language even has its own source level debugger for single stepping through a script and viewing or changing variables on the fly.

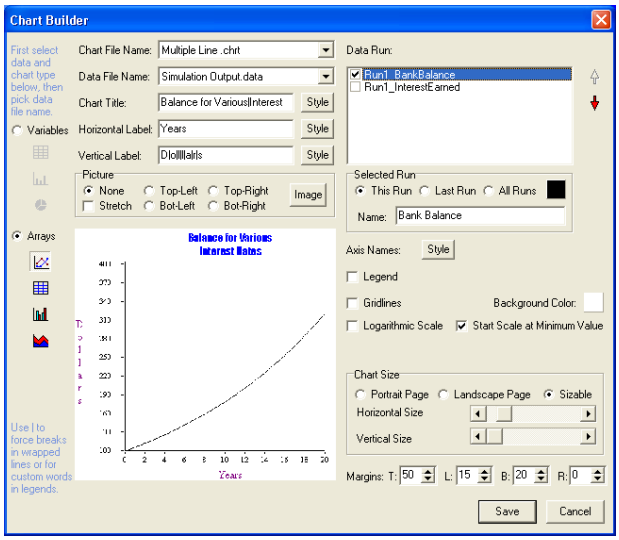
Most users can model and run simulations without writing scripts. A brief explanation of how the simulation script generated by WinA&D works can clarify the process.

First, variables are created and initialized to values derived from the model, the Variables table in the Causal Loop Simulation dialog or from input data files. Next, there's a big loop that runs a scripted routine derived from your equations for each transition line in the model. The loop iterates through each increment in the increment count and stores the variable in a run array created for each variable in the model. Finally, the generated data is output to one or more data files.

Charts & Graphs

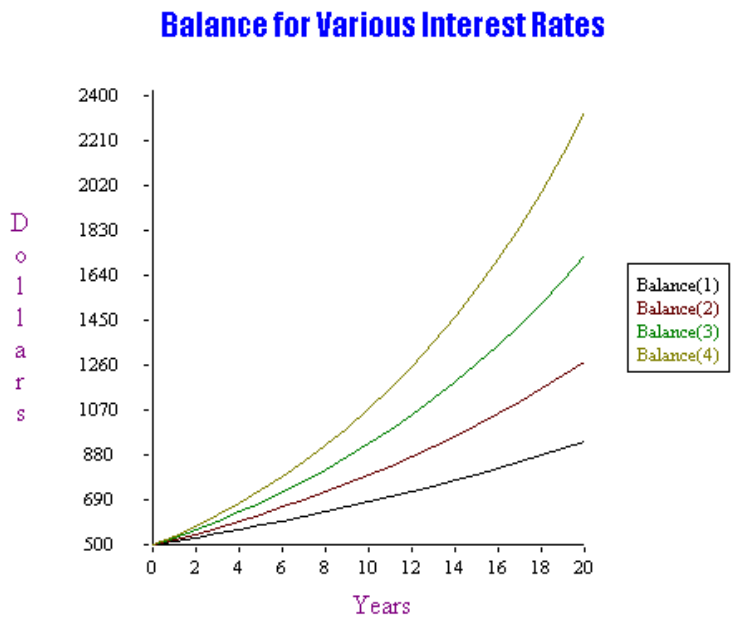
Charts are often used to view the simulated results of a system model that has been parameterized with variable definitions and equations. There are many types of charts for dynamically presenting variable and array data as tables, charts and graphs. Charts are dynamic in the sense that they are driven by data from a data file that when changed will automatically change the chart.

The Chart Builder dialog is used to configure a dynamic chart. The configuration data consists of the chart type, the data values to present and formatting information like titles, labels, colors, text fonts and sizes. Multiple named charts can be configured.



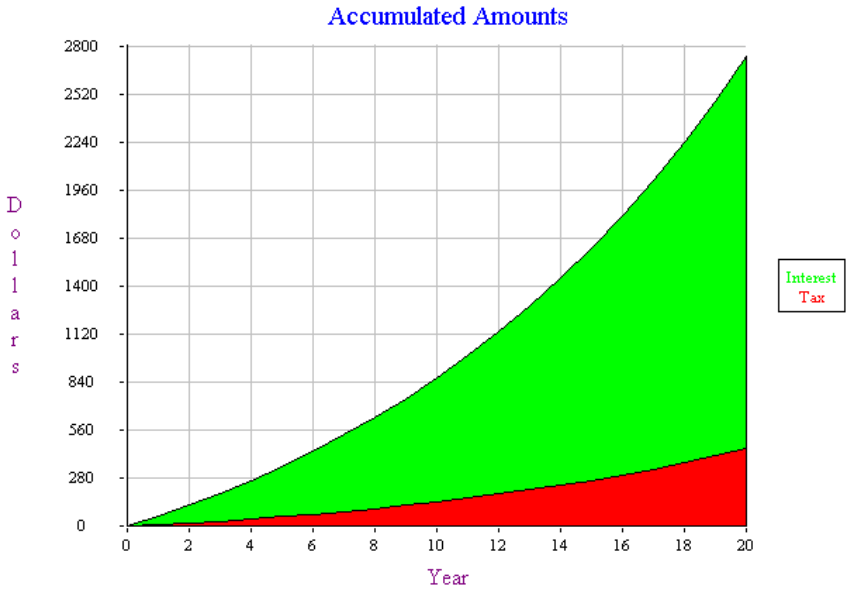
Dialog to Configure a Table, Chart or Graph

In the Bank Account model illustrated above, multiple simulation runs of the model at different interest rates can be shown on the same line graph.



Line Graph of 4%, 6%, 8% and 10% Interest Rates and 20% Tax Rate Over 20 Years

An area chart shows accumulated interest and tax over the years. Tight integration between the model, simulation and charts makes it quick and easy to explore and communicate different scenarios.



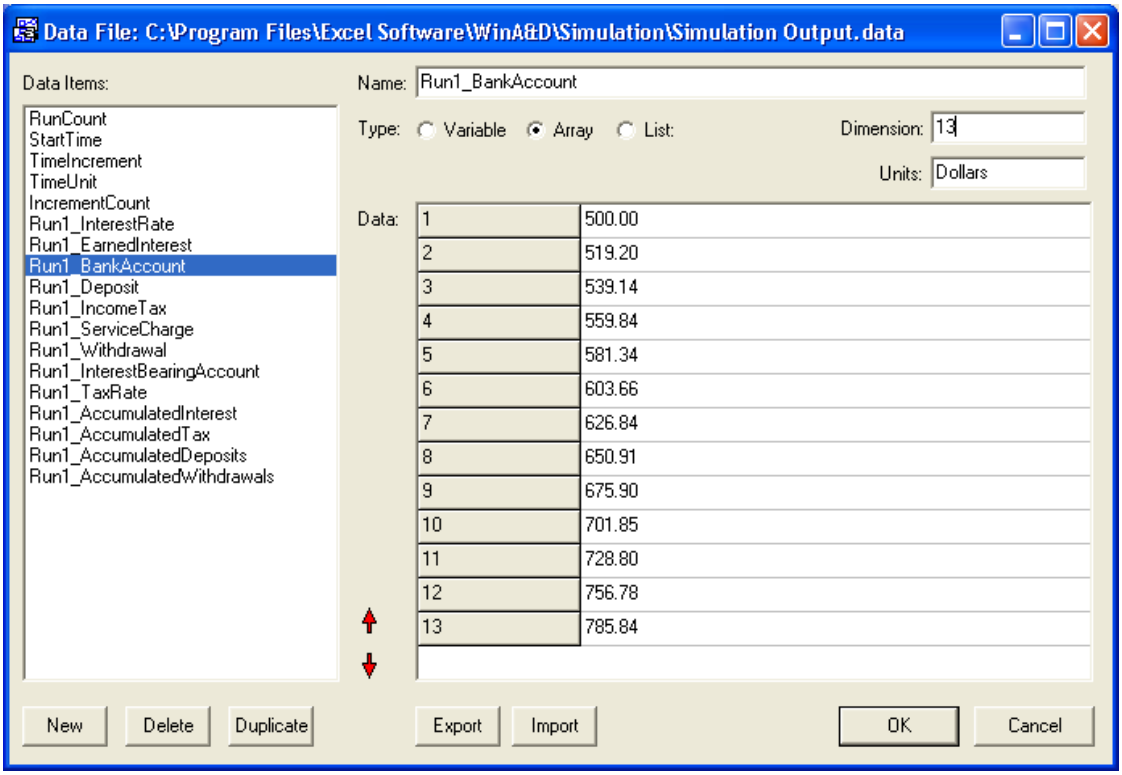
Area Chart of Accumulated Interest and Tax

Input/Output Data Files

A model can get input data from a file or write simulation data to a file. Data files are XML formatted text files used to store, manipulate, import and export data. Data files can be used as input or output in a simulated model, drive dynamic charts or used to read and write data within the scripting environment.

Data files contain a list of named items each of type Variable, Array or List. The output of any simulation produces a few common Variable type items including RunCount and IncrementCount. Each run of the simulation adds a run array for each variable configured in the model. The run array contains the value of that variable at each time increment.

Each simulation run adds another set of run arrays of the form RunX_VariableName.

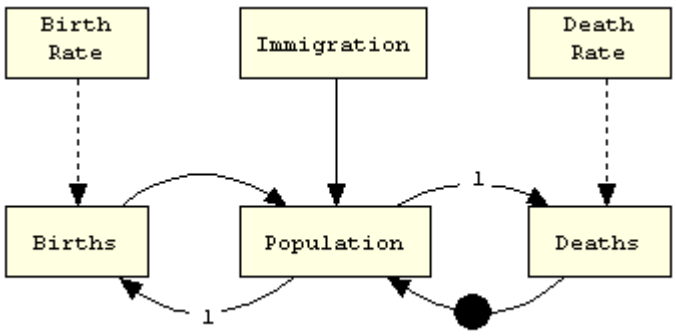


Data File Dialog for Viewing Simulation Output Data

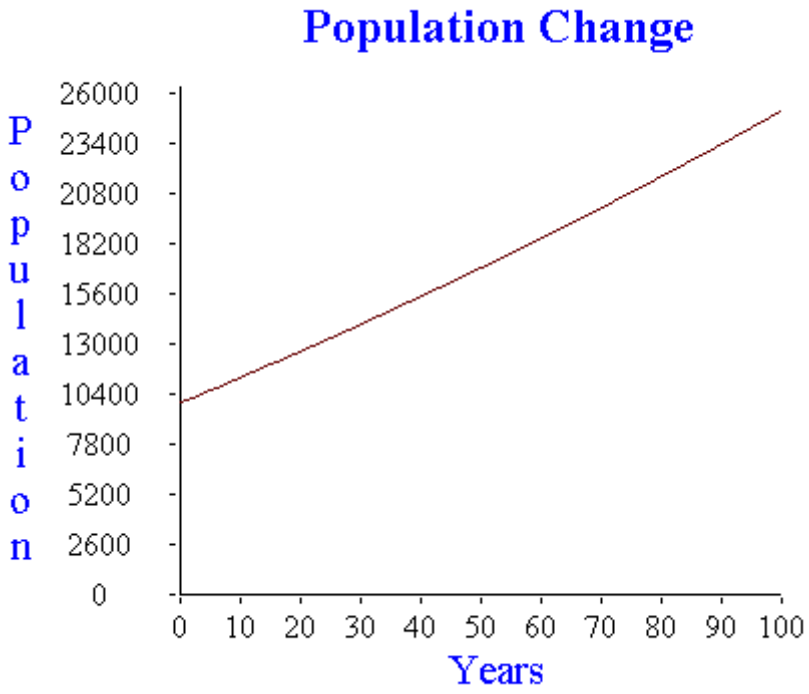
The Import and Export buttons make it easy to share data with another application like a spreadsheet, database or charting tool using tab or comma-delimited files.

Population Example

Population growth is a typical example for demonstrating system models. In this model, the Birth Rate, Immigration and Death Rate variables control how the population changes over time.



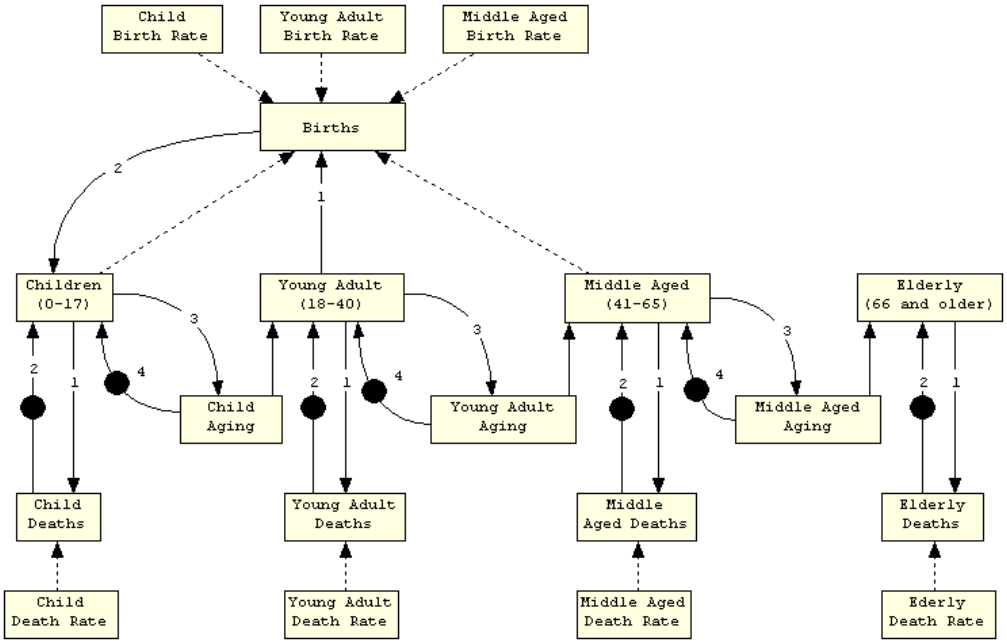
Population Model



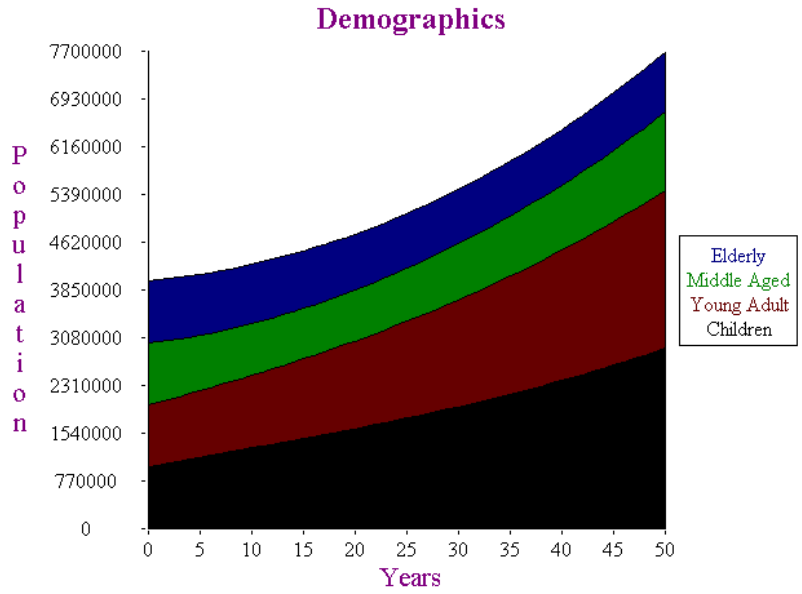
Population Growth Over Time

Demographic Example

In a demographic model, the population is divided into different age groups so the affects of birth rate, death rate and other factors can be more precisely controlled. The model below classifies people as children, young adults, middle aged or elderly based on their age, then allows control variables to changed for each simulation run for each age group.



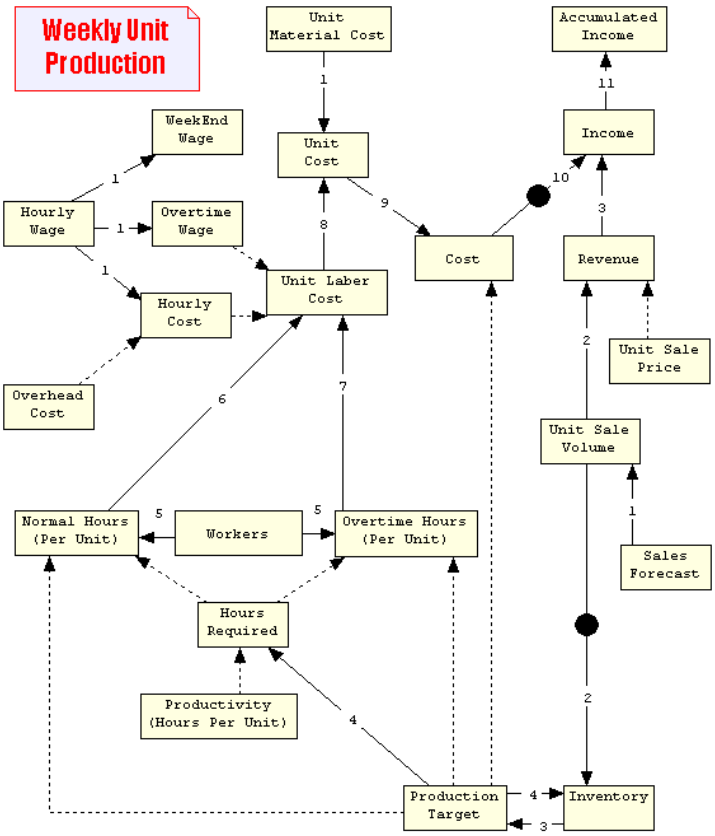
Demographic Models are Useful Predictors of Economic Trends



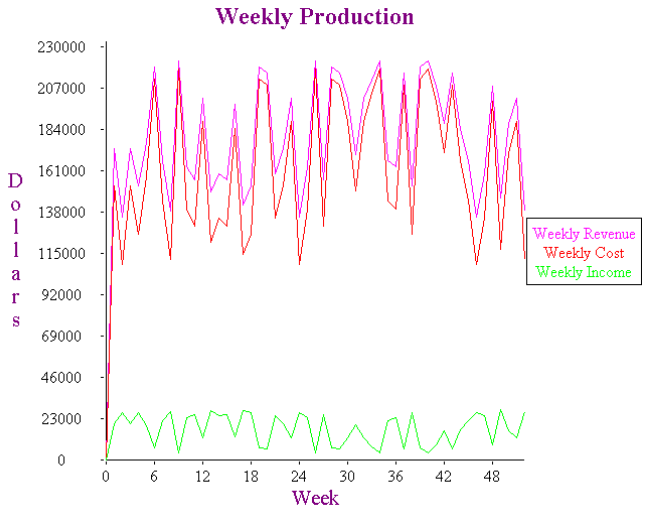
Demographic Simulation Results Shown in Area Chart

Production Example

A manufacturing company needs to track and control costs, revenues, productivity, inventory and make employment projections to maximize income. This model assumes that weekly unit sales will randomly fluctuate +/- 25% from the sales forecast.



Weekly Production Model

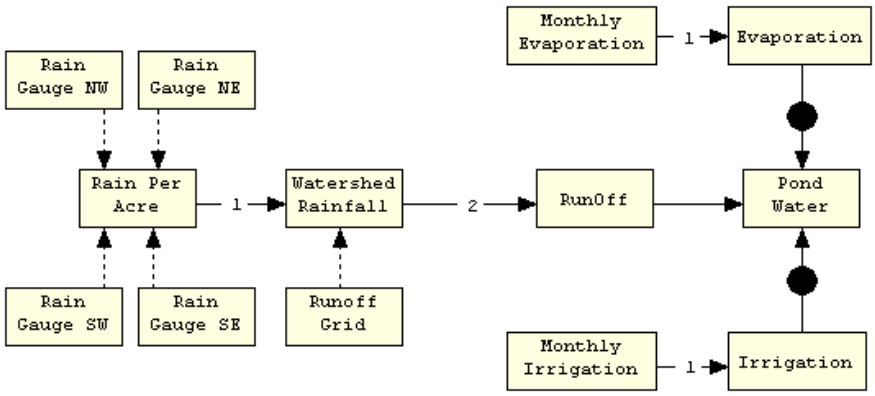


Weekly Graph of Revenue, Cost and Income Based on Control Variables

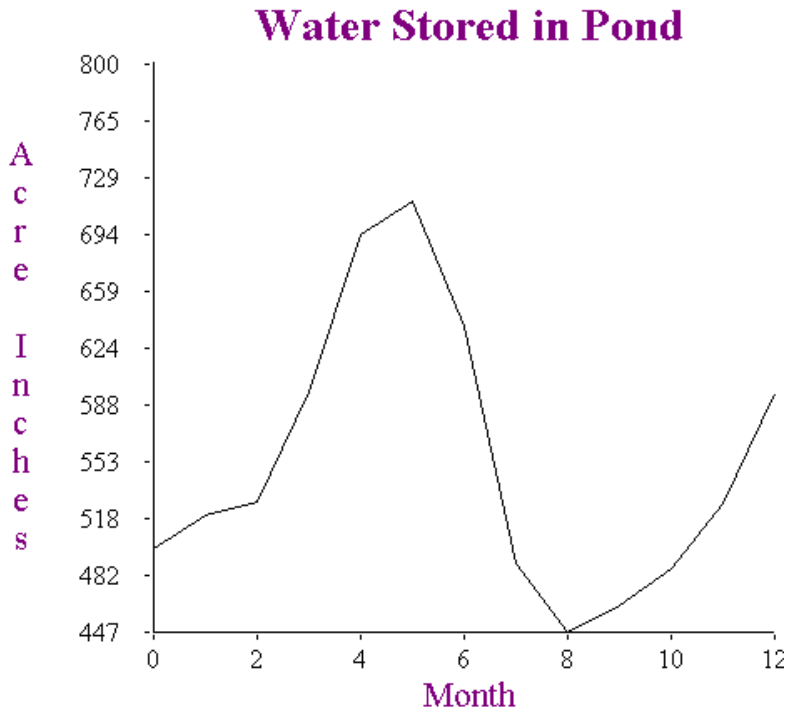
Pond Water Example

This model uses a 10 x 10 acre grid implemented as an array variable for the watershed of an irrigation pond. Rain gauges at each corner measure the rainfall each month. Those measurements are used to estimate the rainfall on each acre and based on its grid number of 0 to 1 will determine how much of that water goes into the pond.

Monthly evaporation and irrigation numbers determine how much water leaves the pond. Rainfall, evaporation and irrigation numbers in this model come from an input data file.



Model of an Irrigation Pond



Monthly Water Stored in Irrigation Pond

Summary

This papers introduced the main concepts of using causal loop diagrams to model dynamic systems by identifying variables and causal affects. The model was parameterized with data and equations and simulated over time. Simulation results can be viewed in a Data File editor, exported to other applications or presented with configurable tables, charts and graphs.

All models, simulations and charts in this paper were created with WinA&D from Excel Software. Causal loop diagrams and charts of simulation data are integrated into other aspects of WinA&D like the report generator and the Project Scrapbook.

About the Author...

Harold Halbleib has a degree in Electrical Engineering and ten years of experience in developing software for real-time, process control systems. He has spent the last 15 years developing Software Engineering tools, training engineers and working with hundreds of software companies around the world. His primary focus has been system modeling & simulation, requirements management, software design, code generation and reengineering, bug tracking, help authoring and programming for Macintosh, Windows and Linux.

Project Perfect is a project management software consulting and training organisation based in Sydney Australia. Their focus is to provide creative yet pragmatic solutions to Project Management issues.

Project Perfect sell “Project Administrator” software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called “Method H”™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au