



Sistema de Apoyo para  
la Simulación de  
Alternativas de Manejo  
Forestal Sostenible

# Manual del Modelizador

28 de enero de 2009



## Índice de contenidos

<b>Introducción .....</b>	<b>1</b>
<b>Proceso de ejecución de modelos.....</b>	<b>1</b>
<b>Origen de datos .....</b>	<b>3</b>
<b>Detalles técnicos .....</b>	<b>3</b>
Lenguaje de programación.....	3
Estructura de los modelos .....	3
Biblioteca de clases .....	3
Navegación.....	4
Seguridad .....	4
<b>Ejemplo de modelo de crecimiento .....</b>	<b>5</b>
<b>Diagrama de entidades .....</b>	<b>8</b>
<b>Referencias .....</b>	<b>11</b>
Programación con VisualBasic.NET .....	11
Ayuda de bibliotecas de clases .....	11
<b>Resumen de características técnicas de los modelos de crecimiento.....</b>	<b>12</b>

## Introducción

Una de las características principales de SIMANFOR es la capacidad para ejecutar modelos de crecimiento sobre inventarios forestales. Los modelos de crecimiento están basados en los modelos elaborados para distintas especies forestales y permiten incluir intervenciones silvícolas, análisis de mortalidad y regeneración, factores de competencia, etc. Estos modelos deben ser programados por los modelizadores, para lo cual es necesario conocer los detalles técnicos de los mismos y cómo son cargados y ejecutados por la aplicación.

Este manual detalla los conceptos y tareas que los modelizadores deben conocer para programar modelos correctamente.

## Proceso de ejecución de modelos

La ejecución de los modelos pasa por varias fases desde que se le indican los árboles que debe procesar hasta que obtiene como resultado un nuevo inventario con los resultados. El diagrama de la página siguiente muestra el flujo de ejecución de los modelos de crecimiento.

Las fases por las que pasan los modelos están indicadas en este diagrama como cajas con borde negro. El significado de cada fase es el siguiente:

**1. Inicialización:**

Permite al modelizador la inicialización de las variables no disponibles en el propio inventario, pero necesarias para la ejecución del modelo. Se ejecuta una vez por pie mayor incluido en el escenario.

**2. Mortalidad:**

Determina si un árbol sobrevive o no al paso del tiempo. Se ejecuta una vez por árbol incluido en el escenario.

**3. Crecimiento:**

Modifica las propiedades del árbol nuevo después de la proyección temporal. Se ejecuta una vez por árbol vivo del escenario.

**4. Masa Incorporada:**

Realiza la inclusión de nuevos árboles al escenario como resultado del nacimiento de estos. Se ejecuta una vez por parcela.

**5. Pre Cálculos:**

Permite al modelo la inicialización de variables utilizadas para los cálculos de pies mayores y parcelas (por ejemplo, diámetros medio/máximo, secciones medias, etc.). Se ejecuta una vez por parcela.

**6. Cálculos Pie Mayor:**

Calcula los datos del árbol que requieren una comparativa con el resto de árboles de la parcela. Se ejecuta una vez por árbol vivo incluido en el escenario.

**7. Cálculos Parcela:**

Calcula los datos de la parcela una vez finalizada la proyección de todos los árboles. Se ejecuta una vez por parcela.

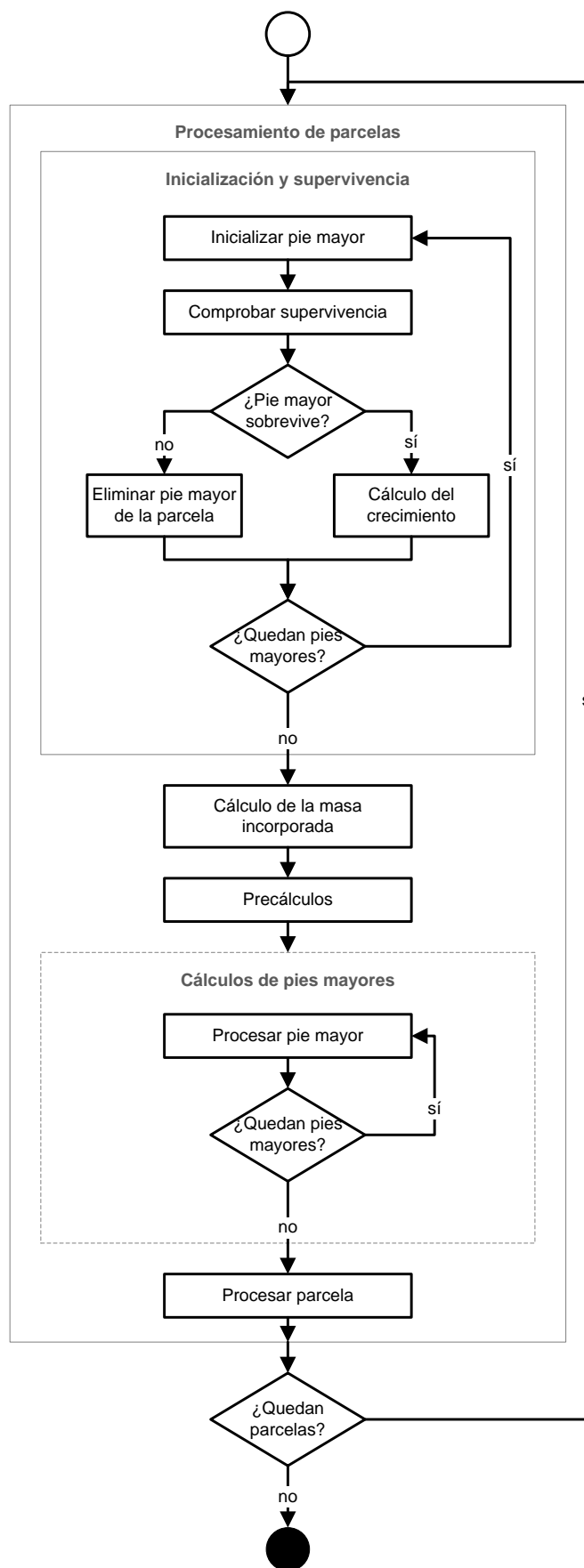


Diagrama de ejecución de los modelos de crecimiento

## Origen de datos

Los modelos de crecimiento trabajan siempre con un modelo de datos similar al SDM, pero distinto en cuanto a las entidades con las que puede trabajar. Esta versión del SDM es bastante más reducida, dado que para la ejecución de modelos no son necesarias muchas de las tablas originales.

Estos inventarios son creados internamente por la aplicación durante la ejecución del modelo y pueden proceder de dos fuentes distintas:

- a. Inventario original, tras aplicar los criterios de filtrado que el usuario haya seleccionado.
- b. Inventario resultante de aplicar el modelo o una corta anteriormente.

En cualquiera de los dos casos, el modelo de crecimiento nunca trabaja con los datos del SDM, sino que siempre trabaja con una versión especial de inventarios que se usa para almacenar temporalmente los datos que se van generando en cada paso.

## Detalles técnicos

### Lenguaje de programación

El lenguaje de programación de los modelos es VisualBasic.NET. Este lenguaje es muy simple y permite un aprendizaje rápido en caso de que no se haya usado antes.

Al final de este documento existe una relación de enlaces sobre la iniciación a la programación de VisualBasic.NET y los detalles técnicos del propio lenguaje.

### Estructura de los modelos

Los modelos de crecimiento no son más que clases (estructuras) que agrupan un conjunto de funciones y procedimientos, que son las que contienen los algoritmos para cada fase. Existe una función o procedimiento para cada fase de las indicadas anteriormente.

Es importante saber que el motor de SIMANFOR es el que se encarga de ejecutar cada una de ellas en cada momento, y que su posición en el código del modelo no tiene relación alguna. Es decir, aunque se recomienda un orden concreto de aparición y definición de cada función o procedimiento, no es estrictamente necesario mantenerlo.

En la siguiente sección puede ver el código de un modelo simple, totalmente operativo y con comentarios en el código para que comprenda los detalles técnicos del mismo.

### Biblioteca de clases

Asociado al lenguaje de programación, existe un conjunto de clases que se pueden usar para definir el tipo de las variables que se usen: números enteros o reales, cadenas de texto, vectores y matrices, etc.

Además, existe una clase denominada Math que agrupa todas las funciones matemáticas y constantes numéricas comunes: número pi y e, funciones de logaritmos, medias aritméticas, trigonométricas, etc.

Al final de este documento existe una relación de enlaces desde los que se pueden consultar los detalles de las bibliotecas de clases y las funciones que contiene la clase Math. En estos enlaces también se pueden consultar ejemplos prácticos para entender el funcionamiento y uso de los mismos.

## Navegación

Las variables usadas en los modelos permiten realizar desplazarse a las distintas entidades del inventario. Por ejemplo, dado un pie mayor, se puede obtener la parcela o el inventario al que pertenece a través de sus propiedades. También se puede recorrer todos los árboles que contiene una parcela a partir de la misma.

Al final de este documento encontrará el diagrama de entidades en el que se indican las propiedades de cada entidad del inventario (el inventario propiamente, las parcelas y los pies mayores), así como qué propiedades permiten navegar hasta otras entidades.

## Seguridad

Los modelos se ejecutan en el servidor bajo una plataforma de seguridad que no permite el acceso a áreas sensibles del sistema como el sistema de archivos o el Registro de Configuración del sistema. Aunque la biblioteca estándar de clases que se usa en los modelos permite usar clases que accedan a estas áreas sensibles, dicha plataforma de seguridad bloquea los accesos, por lo que se garantiza la seguridad del servidor.

Es recomendable que los modelizadores eviten el uso de aquellas clases que pretendan acceder o modificar partes del sistema, y se restrinjan a usar los tipos básicos de la biblioteca de clases y sus operaciones sobre las mismas.

## Ejemplo de modelo de crecimiento

El siguiente ejemplo muestra un ejemplo comentado con el código de un modelo de crecimiento simple en el que se usan las operaciones más habituales en cualquier modelo.

```
Imports System
Imports System.Collections.Generic
Imports Simanfor.Core.EngineModels

' Clase base del modelo.
' Todas las funciones y procedimientos son opcionales. Si se elimina cualquiera
' de ellas, se usará un
' procedimiento o función por defecto que no modifica el estado del inventario.
Public Class Model
    Inherits ModelBase

    ' Declara una variable de modelo, que estará disponible para todas las
    ' funciones y procedimientos
    Private _sumExpan As Single

    ' Procedimiento que permite la inicialización de variables necesarias para
    ' la ejecución del modelo
    Public Overrides Sub Initialize(ByVal tree As PieMayor)
        ' No realiza ninguna operación de inicialización
    End Sub

    ' Función que indica si el árbol sobrevive o no después de <years> años
    ' Devuelve True cuando el árbol sobrevive, o False cuando el árbol muere
    Public Overrides Function Survives(ByVal years As Double, ByVal tree As
PieMayor) As Boolean
        ' Declara una variable local a la función
        Dim threshold As Single

        ' Comprueba que DAP e I_REINEKE no son nulos para realizar las
        ' operaciones
        If tree.DAP.HasValue And tree.Parcela.I_REINEKE.HasValue Then
            ' Usa la clase Math y sus funciones para realizar un cálculo
            ' complejo
            threshold = Math.Pow(Math.Exp((1 / Math.Pi) * years),
(tree.DAP.Value / tree.Parcela.I_REINEKE.Value))
        Else
            threshold = 0
        End If

        ' Decide según el valor calculado si el árbol sobrevive o no
        If threshold < 1 Then
            ' Si EXPAN no es nulo, el árbol computa para la variable _sumExpan
            If tree.EXPAN.HasValue Then
                _sumExpan = _sumExpan + tree.EXPAN.Value
            End If
            Return True
        Else
            Return False
        End If
    End Function

    ' Procedimiento que permite modificar las propiedades del árbol durante su
    ' crecimiento después de <years> años
```

```
Public Overrides Sub Grow(ByVal years As Double, ByVal tree As PieMayor)
    ' Modifica una propiedad para indicar el crecimiento
    If tree.DAP.HasValue Then
        tree.DAP = tree.DAP.Value + years / _sumExpan
    Else
        tree.DAP = years / _sumExpan
    End If
End Sub

' Procedimiento que permite añadir nuevos árboles a una parcela después de
' <years> años
' Todos los árboles nuevos deben añadirse a la lista <list>.
Public Overrides Sub AddTree(ByVal years As Double, ByVal plot As Parcela,
ByVal list As IList(Of PieMayor))
    Dim newTrees As Int32
    newTrees = _sumExpan / list.Count

    Dim i As Int32
    For i = 0 To newTrees
        Dim newTree As PieMayor
        ' Crea un nuevo árbol vacío, sin enlazar con ninguna parcela y con
        ' todos los valores nulos o a cero
        newTree = New PieMayor

        ' Enlazamos el árbol con la parcela
        newTree.Parcela = plot

        ' Establecemos el valor de una propiedad
        newTree.DAP = 50

        ' Agregamos el árbol a la lista
        list.Add(newTree)
    Next i
End Sub

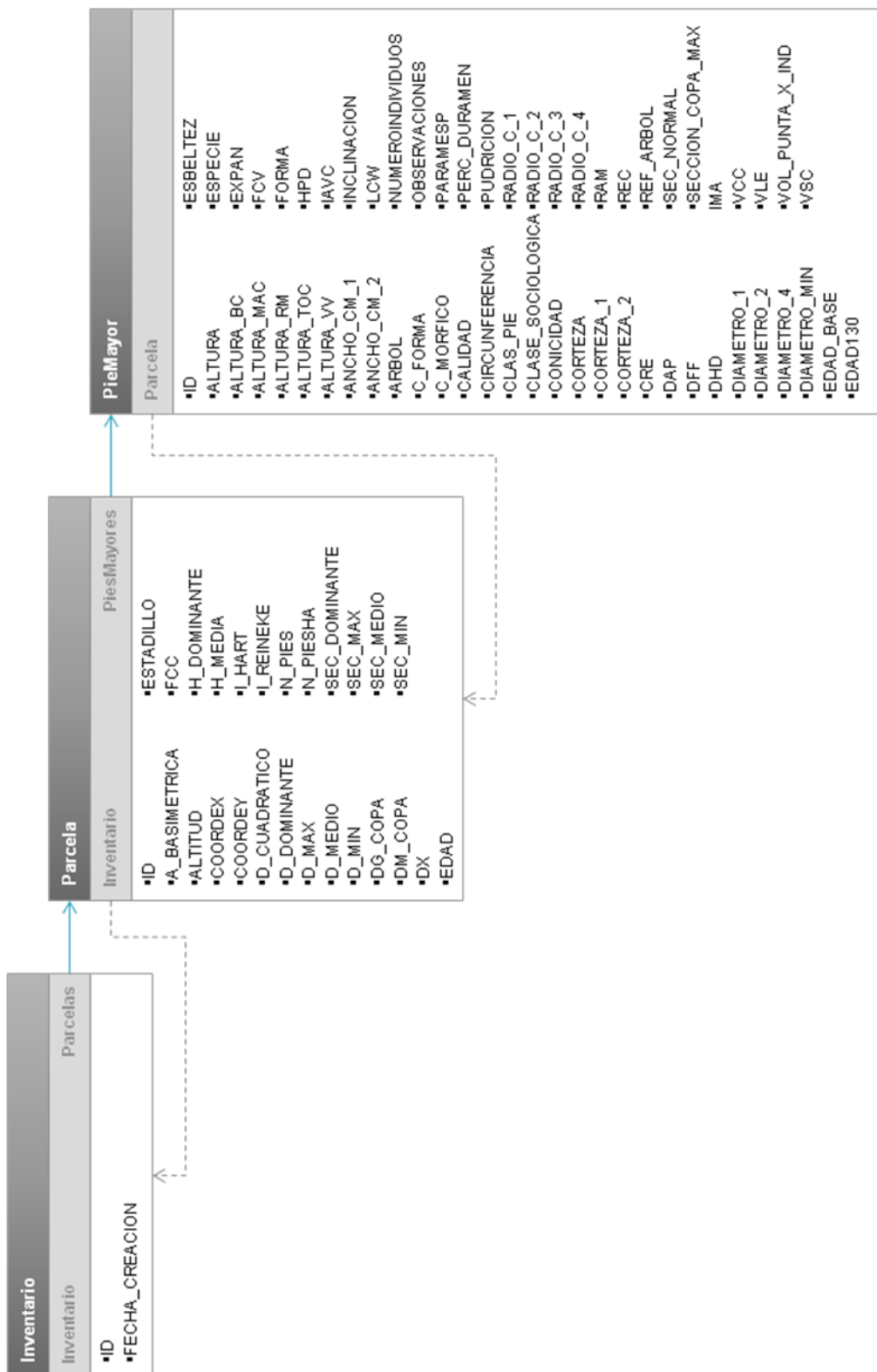
' Procedimiento que realiza todos los cálculos para preparar el
' procesamiento de los árboles y parcelas.
' La colección <variables> se debe usar para almacenar todas las variables y
' valores necesarios para cálculos posteriores.
Public Overrides Sub PreCalculation(ByVal years As Double, ByVal list As
IList(Of PieMayor), ByVal variables As IDictionary(Of String, Object))
    ' Almacena un valor en la colección de variables con el nombre
    ' edad_parcela. Para asignarle un valor,
    ' navega desde el primer elemento de la lista de pies mayores hasta la
    ' parcela a la que pertenece
    variables("edad_parcela") = list(0).Parcela.EDAD
End Sub

' Procedimiento que realiza los cálculos sobre un árbol.
' La colección <variables> contiene las variables previamente almacenadas en
' PreCalculation y puede almacenar nuevas
' para cálculos posteriores.
Public Overrides Sub ProcessTree(ByVal years As Double, ByVal tree As
PieMayor, ByVal variables As IDictionary(Of String, Object))
    ' Recupera el valor anteriormente almacenado en la colección de
    ' variables y lo asigna a una propiedad de la parcela
    tree.EDAD130 = variables("edad_parcela")
End Sub
```

```
' Procedimiento que realiza los cálculos sobre una parcela.  
' La lista <list> contiene los árboles de la nueva parcela.  
' La colección <variables> contiene las variables previamente almacenadas en  
' PreCalculation y puede almacenar nuevas  
' para cálculos posteriores.  
Public Overrides Sub ProcessPlot(ByVal years As Double, ByVal plot As  
Parcela, ByVal list As IList(Of PieMayor), ByVal variables As  
IDictionary(Of String, Object))  
    ' Modifica una propiedad de la parcela  
    If plot.EDAD.HasValue Then  
        plot.EDAD = plot.EDAD.Value + 1  
    End If  
End Sub  
End Class
```

## Diagrama de entidades

El siguiente diagrama presenta las tres entidades con las que trabajan los modelos, sus propiedades y aquellas que permiten navegar entre entidades.



Muchas de las propiedades de estas entidades pueden ser nulas (consultar el SDM para más detalles). Estas propiedades deben tratarse de forma distinta a cualquier otra variable común, por lo que se recomienda seguir estas indicaciones:

- Para obtener el valor actual de la propiedad se debe usar la siguiente sintaxis:

<nombre de la propiedad>.Value

- Toda operación matemática con una propiedad nula puede provocar un error, por lo que se recomienda comprobar dicho valor:

```
If <Entidad>.<Propiedad>.HasValue Then  
  <La propiedad no es nula y se pueden realizar operaciones con ella>  
End If
```

- Para asignar un valor a la propiedad se debe realizar como si se tratara de cualquier otra variable:

```
<nombre de la propiedad> = valor
```

## Referencias

### Programación con VisualBasic.NET

El siguiente enlace muestra la sección de Introducción a VisualBasic.NET, versión 9.0, de MSDN Library. Esta sección incluye los principios del lenguaje, su sintaxis y ejemplos de cómo programar de forma general en este lenguaje.

- **Introducción a VisualBasic.NET 9.0 (.NET Framework 3.5 SP1)**  
**MSDN Library (Español)**  
<http://msdn.microsoft.com/es-es/library/2x7h1hfk.aspx>

### Ayuda de bibliotecas de clases

Los siguientes enlaces muestran las secciones de documentación de la biblioteca estándar de clases y la documentación de Math (también accesible desde el primer enlace), respectivamente. Las clases indicadas en dicha documentación pueden usarse en los modelos, aunque se deben tener en cuenta las limitaciones de seguridad impuestas por el motor de ejecución de modelos.

- **Biblioteca estándar de clases de .NET Framework 3.5 SP1**  
**MSDN Library (Español)**  
<http://msdn.microsoft.com/es-es/library/ms229335.aspx>
- **Referencia de la clase Math**  
**MSDN Library (Español)**  
<http://msdn.microsoft.com/es-es/library/system.math.aspx>

## Resumen de características técnicas de los modelos de crecimiento

<b>Lenguaje de programación</b>	VisualBasic.NET 2008
<b>Versión de .NET Framework</b>	Microsoft .NET Framework 3.5 SP1
<b>Zona de seguridad de ejecución</b>	Zona de Internet (privilegios mínimos)
<b>Ensamblados incluidos</b>	mscorlib Simanfor.Core.EngineModels
<b>Espacios de nombres por defecto</b>	System System.Collections.Generics Simanfor.Core.EngineModels
<b>Tipos base usados por defecto</b>	System.Double System.Object System.String System.Collections.Generics.IDictionary<> System.Collections.Generics.IList<>
<b>Tipos propios usados por defecto</b>	Simanfor.Core.EngineModels.PieMayor Simanfor.Core.EngineModels.Parcelas