# MODGEN ET L'APPLICATION RISKPATHS DU POINT DE VUE DU DÉVELOPPEUR DE MODÈLES

Martin Spielauer

Statistique Canada – Division de la modélisation
Immeuble R.-H.- Coats, 24-O
Ottawa, K1A 0T6

martin.spielauer@statcan.gc.ca

RiskPaths est un modèle de microsimulation simple, à temps continu, basé sur les cas et à risques concurrentiels. Ce modèle est principalement utilisé comme outil d'enseignement, pour introduire la microsimulation à des spécialistes des sciences sociales et démontrer comment les modèles de microsimulation dynamique peuvent être programmés efficacement en utilisant le langage Modgen.

Modgen est un langage de programmation générique des modèles de microsimulation développé et maintenu à Statistique Canada.

RiskPaths, comme le langage de programmation Modgen et d'autres documents connexes, sont disponibles à www.statcan.gc.ca/microsimulation/modgen/modgen-fra.htm

## 1 Introduction

Dans le présent document, nous examinons le progiciel de développement de modèle de microsimulation Modgen et l'application RiskPaths de Modgen du point de vue du développeur de modèles. Nous décrivons d'abord l'environnement de programmation Modgen, puis nous discutons des concepts de base du langage Modgen et du code de RiskPaths. L'utilisation de Modgen ne nécessitant que des compétences de programmation modestes, elle permet aux spécialistes des sciences sociales, moyennant une certaine formation, de créer leurs propres modèles sans devoir recourir à des programmeurs professionnels. Cet exercice est possible parce que Modgen cache les mécanismes sous-jacents, tels que la mise en file d'attente des événements et la création automatique d'un

modèle autonome doté d'une interface visuelle complète, y compris la gestion des scénarios et la documentation du modèle (présentée dans le chapitre précédent). Par conséquent, les développeurs de modèles peuvent se concentrer sur le code propre au modèle : la déclaration des paramètres, les états définissant les acteurs simulés et les événements modifiant les états. Le codage à haut rendement s'étend aussi aux données de sortie du modèle. Modgen comprend un langage puissant pour traiter les totalisations en temps continu. Ces totalisations sont créées à la volée durant l'exécution des simulations et la programmation pour les produire ne requiert habituellement que quelques lignes de codes par tableau. Modgen est également doté d'un mécanisme intégré pour l'estimation de la variation Monte Carlo pour toute cellule de n'importe quel tableau, sans aucune programmation requise de la part du développeur du tableau.

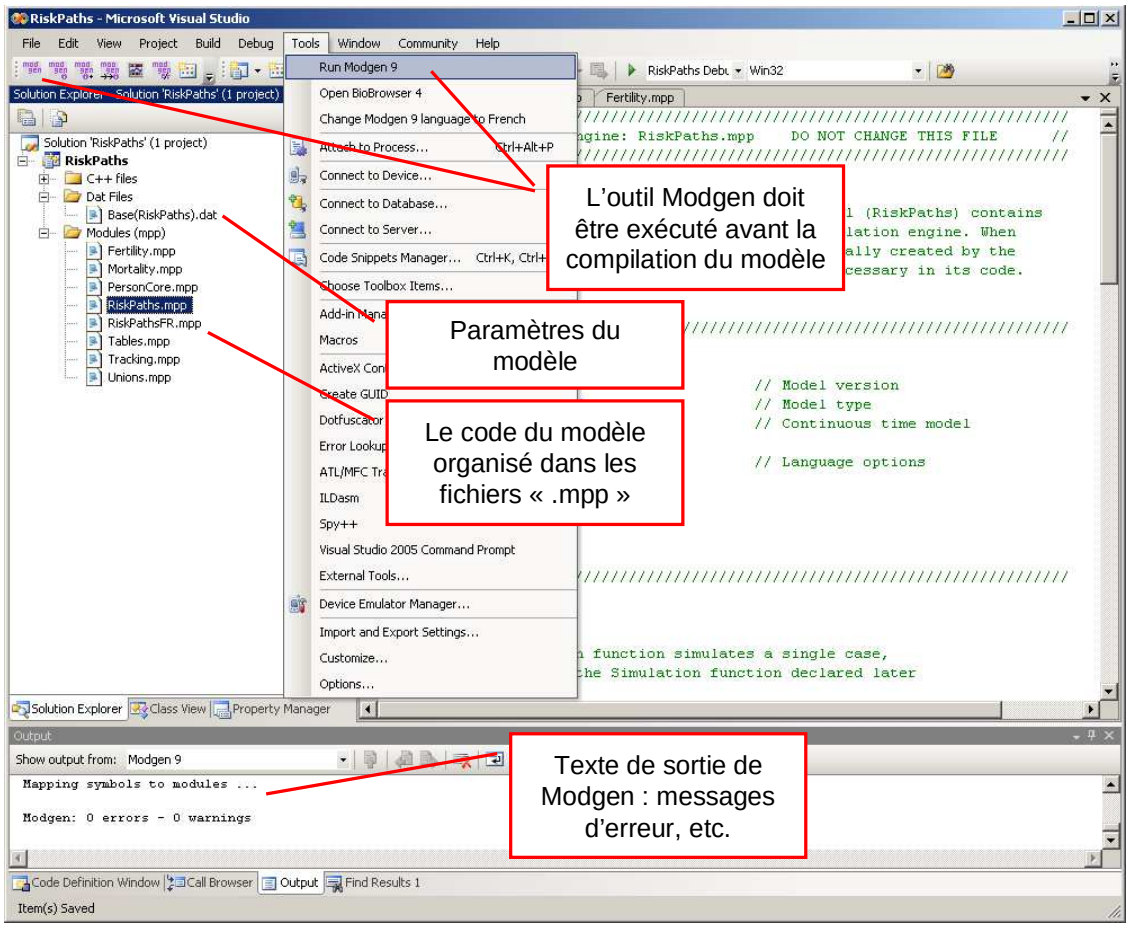
Étant un modèle simple, RiskPaths n'utilise pas la gamme complète d'éléments de langage et de capacités de Modgen. La discussion qui suit n'est pas destinée à remplacer la documentation existante sur Modgen, telle que le guide du développeur. Cependant, en présentant ici les principaux concepts de la programmation Modgen, nous souhaitons vous aider à mettre en route le développement de modèles et à entreprendre une exploration plus approfondie.

## 2 L'environnement de programmation Modgen

Lorsqu'il est installé sur un ordinateur, Modgen s'intègre dans l'environnement Microsoft Visual Studio C++ (requis). Les composantes visuelles de Modgen sont des barres d'outils distinctes, ainsi que des éléments supplémentaires sous les menus « Tools » et « Help » de Visual Studio. Modgen figure aussi comme une option dans la boîte de dialogue des fichiers pour créer un nouveau projet, ainsi que dans celle pour ajouter un fichier à un projet existant.

La Figure 1 représente une vue d'écran de l'interface de programmation telle qu'elle apparaît après l'ouverture de l'application « RiskPaths.sln » de Modgen. La barre d'outils Modgen comprend des icônes pour l'exécution de Modgen, l'accès aux fichiers d'aide, l'ouverture de l'outil BioBrowser et le changement de la langue (de l'anglais au français et inversement).

**Figure 1 : Interface de programmation**




Le code de Modgen est réparti dans plusieurs fichiers, portant chacun l'extension .mpp. Comme le montre la fenêtre Solution Explorer (Figure 1), RiskPaths comprend huit fichiers .mpp groupés dans le répertoire « Models (mpp) ». Ces fichiers sont ceux qui sont essentiels à l'exécution de RiskPaths, c'est-à-dire les fichiers contenant tout le code Modgen écrit par le développeur du modèle.

Lorsque l'outil Modgen (auquel on peut avoir accès à partir de la barre d'outils ou à partir du premier élément du menu « Tools ») est appelé, ces fichiers .mpp sont traduits en code C++. Donc, Modgen agit comme un précompilateur qui crée un fichier de code source .cpp pour chaque fichier .mpp et plaçe les fichiers .cpp résultant dans le répertoire « C++ Files ». L'outil Modgen ajoute aussi des composantes de code C++ indépendantes du modèle au répertoire « C++ Files »; ces fichiers supplémentaires[1] ne doivent pas être modifiés par le développeur

[1] ACTORS.CPP, ACTORS.H, app.ico, model.h, model.RC, PARSE.INF, TABINIT.CPP, TABINIT.H.

du modèle et sont essentiels à l'utilisation du compilateur C++ pour construire l'application Modgen.

Les paramètres du modèle sont contenus dans un ou plusieurs fichiers .dat classés dans un répertoire étiqueté « Scenarios ». Ces fichiers sont chargés au moment de l'exécution et contiennent les valeurs réelles attribuées aux paramètres.

Durant l'exécution de l'outil Modgen, celui-ci — comme le compilateur C++ — produit un journal qui est affiché dans la fenêtre Output. Tout message d'erreur est également affiché dans cette fenêtre et en cliquant sur un message d'erreur particulier, vous arrivez directement au code Modgen correspondant qui a produit l'erreur.

Deux étapes sont nécessaires pour créer une application Modgen dans l'environnement Visual Studio. Premièrement, Modgen doit traduire le code Modgen qui figure dans les fichiers .mpp; il faut pour cela appeler l'outil Modgen. Deuxièmement, l'application C++ résultante doit être construite et démarrée. Cela peut se faire en une étape en sélectionnant « Start Debugging » dans le menu « Debug » ou en cliquant sur l'icône correspondant de la barre d'outils.

## 3 Concepts de base de Modgen

**Acteur :** Un acteur est l'entité dont la vie est simulée dans un modèle Modgen. Il s'agit souvent d'une personne, bien que cela ne soit pas une exigence — d'autres modèles ont été développés pour simuler des logements ou des professions comme acteurs. Néanmoins, dans RiskPaths, l'acteur est une personne ou, plus précisément, une femme (puisqu'il s'agit d'un modèle conçu pour étudier le fait de ne pas avoir d'enfants).

**État :** Les états décrivent les caractéristiques des acteurs d'un modèle. Certains états peuvent être continus, comme l'âge, tandis que d'autres sont catégoriques, comme le sexe. Pour les états catégoriques, les catégories ou les niveaux réels sont définis à l'aide de la commande **classification** de Modgen.

Dans l'ensemble, il existe deux grands types d'états dans Modgen — les **états simples** et les **états dérivés**, qui sont tous deux utilisés dans RiskPaths et qui sont tous deux déclarés dans une déclaration d'acteur. Un état simple est un état dont la valeur peut être initialisée et modifiée par le code que crée un développeur de modèle. Les états simples sont modifiés par des événements explicitement déclarés. Un état dérivé, par ailleurs, est un état dont la valeur est donnée sous forme d'une expression qui est normalement dérivée d'après, ou basée sur, d'autres états. Les valeurs d'un état dérivé sont tenues à jour automatiquement par Modgen tout au long de l'exécution d'une simulation. Un autre concept utile de Modgen est l'état dérivé autoplanifié. Il s'agit d'un état qui change dans une séquence temporelle prédéfinie, tel que integer_age, un état de RiskPaths qui change à chaque anniversaire.

**Événement :** Dans Modgen, la simulation a lieu par l'exécution des événements. Un événement comprend deux fonctions : une fonction temporelle pour déterminer le moment de la prochaine occurrence de l'événement et une fonction d'exécution pour déterminer les conséquences de la survenue de l'événement. RiskPaths comprend plusieurs événements, y compris un événement de mortalité, des événements de formation et de dissolution d'une union et un événement de première grossesse.

**Paramètre :** Les paramètres sont utilisés pour donner à l'utilisateur du modèle un certain degré de contrôle sur les simulations qu'il exécute. La capacité de modifier divers risques ou probabilités qui ont une incidence sur divers aspects de la simulation permet d'étudier différents scénarios. Les paramètres peuvent posséder de nombreuses dimensions (telles que l'âge, le sexe et l'année) et sont sauvegardés dans des fichiers de données .dat. RiskPaths contient un fichier de paramètres, Base(RiskPaths).dat, qui contient les valeurs des paramètres telles que les probabilités de décès selon l'âge et les risques d'une première grossesse selon le groupe d'âge. Les modèles plus complexes contiennent habituellement plus d'un fichier .dat.

**Tableau :** Une fonction de totalisation croisée puissante est intégrée dans Modgen pour présenter les résultats agrégés sous la forme de tableaux. Une déclaration de tableau comprend deux éléments centraux, à savoir ses dimensions de capture (qui définissent quand un acteur entre dans une cellule et quand il en sort) et sa dimension d'analyse (qui enregistre ce qui se passe pendant que l'acteur est dans la cellule). Pendant l'exécution de simulations, les totalisations destinées à remplir un tableau sont produites à la volée, de sorte qu'il n'est pas nécessaire de créer de grands fichiers temporaires et d'y inscrire les données en vue de la production subséquente de rapports. Plusieurs exemples de déclaration de tableau sont présentés plus loin dans le document pour le modèle RiskPaths.

## 4 Organisation des fichiers

Le code Modgen de RiskPaths est réparti entre huit fichiers .mpp distincts, tandis que toutes les valeurs des paramètres RiskPaths (parce que dernier est un modèle simple) sont contenues dans un seul fichier .dat. En principe, un développeur de modèle est entièrement libre de décider comment répartir le code Modgen dans les différents fichiers, mais une structure modulaire telle que celle de RiskPaths, est recommandée.

**Figure 2 : Organisation des fichiers de RiskPaths**

| Modules généraux | Nom du fichier |
|---|---|
| Moteur de simulation | RiskPaths.mpp |
| Fichier de base des acteurs | PersonCore.mpp |
| Définitions des tableaux | Tables.mpp |
| Dépistage des sorties | Tracking.mpp |
| Traductions en français | RiskPathsFR.mpp |

| Modules de comportement | Nom du fichier |
|---|---|
| Mortalité | Mortality.mpp |
| Fécondité | Fertility.mpp |
| Formations et dissolutions d'unions | Unions.mpp |

| Fichier des paramètres | Nom du fichier |
|---|---|
| Paramètres du scénario de référence | Base(RiskPaths).dat |

Il convient de souligner que les fichiers de code .mpp contiennent souvent des commentaires qui ressemblent à des étiquettes. Ces commentaires sont placés à côté des déclarations des symboles, tels que les états, les niveaux d'état, les paramètres, les tableaux et les dimensions de tableau. Modgen interprète en fait ces commentaires comme des étiquettes et les utilise subséquemment quand les tableaux ou les paramètres sont affichés dans l'interface visuelle de Modgen. Ces étiquettes sont alors utilisées dans le fichier d'aide encyclopédique du modèle qui est produit automatiquement. Les commentaires du code qui sont utilisés comme des étiquettes commencent par un identificateur de langue de deux caractères, par exemple :

```
//EN Union status
```

De nombreux commentaires de ce genre figurent dans les exemples de code qui suivent pour RiskPaths.

Pour des descriptions plus détaillées des modules, des fonctions et des événements, des notes utilisant la syntaxe qui suit peuvent également être placés dans le code.

```
/*NOTE(Person.Finish, EN)
      The Finish function terminates the simulation of an actor.
*/
```

En plus de renseigner sur le code, ces notes sont utilisées dans le fichier d'aide encyclopédique produit automatiquement.

### 4.1 RiskPaths.mpp (le fichier de simulation principal)

Ce fichier contient le code essentiel à la définition du type de modèle (p. ex. orienté cas, temps continu) ainsi que le moteur de simulation, c'est-à-dire le code qui exécute la simulation complète. Parce que RiskPaths est un modèle orienté cas, le code du moteur de simulation est

itéré sur tous les cas et traite les files d'événement de chaque cas. Le fichier identifie aussi les langues utilisées dans le modèle. Le code contenu dans ce fichier est en grande partie indépendant du modèle à l'intérieur d'une classe de modèles (p. ex. temps continu, orienté cas) et une version de ce fichier est fournie automatiquement quand le guide intelligent intégré dans Modgen est utilisé pour lancer un nouveau projet Modgen.

Pour le développement de notre modèle RiskPaths de cohorte en temps continu orienté cas avec un acteur « Person », le code fourni par le guide intelligent ne nécessite que très peu de modifications. Le code complet de ce fichier .mpp a moins d'une page de long.

### 4.2 PersonCore.mpp

Le seul acteur dans RiskPaths est une personne. Dan le fichier PersonCore.mpp, nous avons placé le code qui fait partie de la déclaration de l'acteur, mais qui n'est pas directement relié à un comportement spécifique. Le fichier contient deux horloges d'âge définies comme des états autoplanifiés (integer_age et age_status) et deux fonctions d'acteur, Start() et Finish(), qui sont exécutées à la création d'un acteur et à son décès, respectivement.

Dans la fonction Start(), nous initialisons le temps et l'âge des états à 0. Les deux états sont créés et tenus à jour automatiquement par Modgen et ne peuvent être modifiés que dans la fonction Start(). Leur type dépend du type de modèle; comme RiskPaths est un modèle en temps continu, le temps et l'âge sont des états continus.

La fonction Finish() doit être appelée au moment de l'événement de décès d'un acteur. Son rôle est d'éliminer l'acteur des tableaux et de la simulation, et de récupérer toute mémoire informatique utilisée par l'acteur.

Tous les états et fonctions d'acteur sont décrits dans un bloc « actor Person { }; ». Pour permettre que l'organisation du code en fonction de divers domaines de la trajectoire de vie soit modulaire, il peut y avoir plusieurs blocs d'acteur dans un projet, habituellement un pour chaque fichier de comportement.

La première section de code de ce module contient trois types de définitions. Nous commençons par définir un intervalle de vie range LIFE.

```
range LIFE        //EN Simulated age range
{
      0,100
};
```

Range est un type Modgen qui définit un intervalle de valeurs entières. RiskPaths limite l'intervalle d'âge possible des personnes à 100 ans. Ce type est utilisé pour déclarer un état dérivé contenant l'âge d'une personne en années complètes.

Le deuxième type de définition est utilisé pour diviser les âges continus en intervalles d'âge de 2,5 ans en commençant à l'âge de 15 ans.

```
partition AGEINT_STATE        //EN 2.5 year age intervals
{
      15, 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, 40
};
```

La troisième définition est une classification des types d'union. En général, si un intervalle (range), une partition ou une classification est utilisée dans plusieurs fichiers, il est bon de la définir dans le fichier de base de l'acteur.

```
classification UNION_STATE                //EN Union status
{
      US_NEVER_IN_UNION,                  //EN Never in union
      US_FIRST_UNION_PERIOD1,             //EN First union < 3 years
      US_FIRST_UNION_PERIOD2,             //EN First Union > 3 years
      US_AFTER_FIRST_UNION,               //EN After first union
      US_SECOND_UNION,                    //EN Second union
      US_AFTER_SECOND_UNION               //EN After second union
};
```

Dans le segment de code qui suit, nous déclarons deux états d'acteur dérivés et deux fonctions. Les états dérivés des intervalles de temps sont utilisés pour modifier les valeurs des paramètres qui varient en fonction du temps. Dans notre modèle, integer_age est nécessaire parce que les risques de mortalité dépendent de l'âge en années, tandis que age_status intervient parce que les risques de base pour la première grossesse et la formation d'une première union sont modélisés de manière à ce qu'ils changent par intervalles de 2,5 ans après le 15$^{e}$ anniversaire.

Tant integer_age que age_status doit être tenu à jour au cours de la simulation. Le concept Modgen d'état dérivé nous permet de les tenir à jour automatiquement. Tous deux sont dérivés de l'état « âge » (qui est un état particulier produit et tenu à jour automatiquement par Modgen). Afin de diviser l'âge en intervalles de temps définis dans la partition AGEINT_STATE, nous utilisons la fonction Modgen self_scheduling_split. Le deuxième état dérivé, integer_age, peut être obtenu directement en utilisant la fonction Modgen self_scheduling_int. Pour être certain que sa valeur reste dans l'intervalle de valeurs possibles de LIFE, nous le convertissons au type LIFE, ce qui est effectué par la macro Modgen COERCE.

```
actor Person
{
      //EN Current age interval
      int age_status = self_scheduling_split(age, AGEINT_STATE);

      //EN Current integer age
```

```
	LIFE integer_age = COERCE( LIFE, self_scheduling_int(age) );

	//EN Function starting the life of an actor
	void Start();

	//EN Function finishing the life of an actor
	void Finish();
}
```

Le code qui reste dans ce module correspond à l'exécution des fonctions Start() et Finish(). La fonction Finish() est laissée vide car nous n'avons besoin d'aucune autre action que celle exécutée automatiquement par Modgen au moment du décès d'un acteur.

```
void Person::Start()
{
	// Age and time are variables automatically maintained by
	// Modgen. They can be set only in the Start function
	age = 0;
	time = 0;
}

/*NOTE(Person.Finish, EN)
	The Finish function terminates the simulation of an actor.
*/
void Person::Finish()
{
	// After the code in this function (if any) is executed,
	// Modgen removes the actor from tables and from the simulation.
	// Modgen also recuperates any memory used by the actor.
}
```

### 4.3 Fichiers de comportement

Dans RiskPaths, nous distinguons trois groupes de comportements : la mortalité, la fécondité et la formation/dissolution d'une union. Par conséquent, nous avons réparti le code dans trois fichiers .mpp : Mortality.mpp, Fertility.mpp et Unions.mpp. Chaque fichier de comportement est habituellement subdivisé en trois sections :

- déclaration des paramètres (incluant tous types de définitions qui sont requises pour commencer);
- déclarations des états d'acteur et des événements;
- exécution des événements.

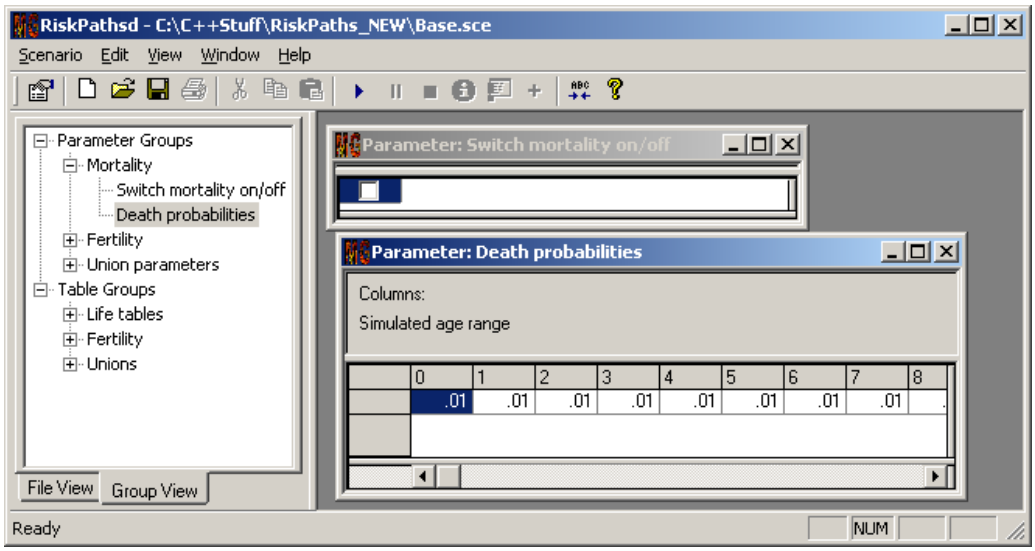
### 4.3.1 Mortality.mpp

Ce fichier définit l'événement de mortalité qui termine la vie de l'acteur simulé. Mortality.mpp est un module de comportement typique et nous suivons une présentation standard du code : déclarations des paramètres (avec les définitions des types), déclarations des acteurs et exécution des événements.

#### Déclarations des paramètres

La mortalité est paramétrisée par les probabilités de décès selon l'âge; donc, la probabilité de survivre une année supplémentaire change à chaque anniversaire. Nous introduisons aussi un paramètre qui nous permet de « désactiver » la mortalité. Quand il est utilisé, chaque acteur atteint l'âge maximal de 100 ans (ce qui peut être utile pour certains types d'analyse de la fécondité). La figure 3 montre les tableaux des paramètres de mortalité de l'application RiskPaths.

**Figure 3 : Paramètres de la mortalité**




Les paramètres sont déclarés dans un bloc de code « parameters {…}; ». Modgen permet d'utiliser les types numériques standard de C++, tels que int, long, float, double ou boolean (« logical » dans la terminologie de Modgen), ainsi que les types range (intervalle), partition et classification particuliers à Modgen qui ont été introduits dans PersonCore.mpp. La dimensionnalité des paramètres dans le modèle RiskPaths est définie par les classifications et les intervalles. Le code qui suit produit pour RiskPaths les paramètres illustrés à la figure 3. Pour les probabilités annuelles de décès, nous utilisons l'intervalle LIFE qui a été défini dans PersonCore.mpp. L'instruction (**parameter_group**) regroupe les deux paramètres de mortalité

afin de fournir une liste de sélection hiérarchique ordonnée dans l'interface utilisateur (de nouveau, comme illustré à la figure 3).

```
parameters
{
      logical CanDie;               //EN Switch mortality on/off
      double  ProbMort[LIFE];       //EN Death probabilities
};

parameter_group P01_Mortality       //EN Mortality
{
      CanDie, ProbMort
};
```

**Déclarations des acteurs**

Les acteurs sont décrits par des états qui sont modifiés par des événements. Les états peuvent être continus (entiers ou réels) ou catégoriques. Dans le module de la mortalité, l'état d'intérêt est le fait qu'une personne est en vie ou non, si bien qu'il est catégorique par nature. Les niveaux d'un état catégorique sont définis au moyen de la commande **classification** de Modgen.

Nous déclarons un état life_status de type LIFE_STATE, qui est initialisé avec LS_ALIVE à la naissance et fixé à LS_NOT_ALIVE par l'événement de décès. L'initialisation de tous les états en attribuant la valeur initiale est une bonne pratique. Cependant, chaque valeur initiale doit être comprise entre des parenthèses, c'est-à-dire {} — sinon, l'état est exécuté comme un état dérivé.

```
classification LIFE_STATE                       //EN Life status
{
      LS_ALIVE,                                 //EN Alive
      LS_NOT_ALIVE                              //EN Dead
};

actor Person


{
      LIFE_STATE life_status = {LS_ALIVE};      //EN Life Status
      event timeDeathEvent, DeathEvent;         //EN Death Event
};
```

Les événements sont déclarés dans le bloc actor Person {..} en utilisant le mot-clé **event**. Tous les événements consistent en une fonction qui produit le temps de l'événement suivant et une fonction contenant le code décrivant les conséquences de l'événement.

**Exécution d'un événement**

Quand l'option de mortalité est activée, la fonction timeDeathEvent produit un temps aléatoire basé sur le paramètre de mortalité pour l'année d'âge donnée. Afin d'obtenir des durées aléatoires d'après les probabilités, nous supposons que les risques de mortalité sont constants durant chaque période, c'est-à-dire entre les anniversaires. (Fait exception la probabilité de décès qui est égale à 1, ce qui entraîne le décès immédiatement après le début de l'année d'âge). Il convient de souligner que tout temps ultérieur à l'anniversaire suivant entraînera la précédence de l'événement d'anniversaire sur l'événement de mortalité; autrement dit, l'événement d'anniversaire censurera l'événement de mortalité.

```
TIME Person::timeDeathEvent()
{
	TIME event_time = TIME_INFINITE;
	if (CanDie)
	{
		if (ProbMort[integer_age] >= 1)
		{
			event_time = WAIT(0);
		}
		else
		{
			event_time = WAIT(-log(RandUniform(3)) /
					-log(1 - ProbMort[integer_age]));
		}
	}
	// Death event can not occur after the maximum duration of life
	if (event_time > MAX(LIFE))
	{
		event_time = MAX(LIFE);
	}
	return event_time;
}
```

La fonction d'exécution de l'événement DeathEvent est simple. Elle donne à life_status la valeur LS_NOT_ALIVE et appelle la fonction Finish(), qui élimine l'acteur de la simulation et récupère tout espace mémoire utilisé par cet acteur.

```
void Person::DeathEvent()
{
	life_status = LS_NOT_ALIVE;
	Finish();
}
```

### 4.3.2 Fertility.mpp

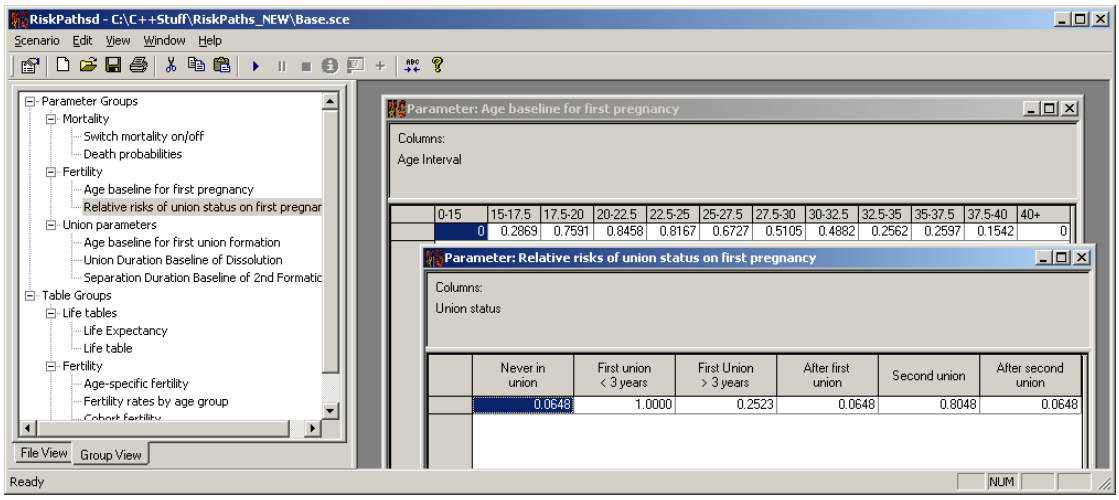
Ce fichier définit et exécute l'événement de première grossesse. Comme nous nous servons de RiskPaths pour étudier l'absence d'enfants, nous ne simulons aucun autre événement relié à la

fécondité. Fertility.mpp est un module de comportement et, de nouveau, nous adoptons la même présentation standard du code : définitions des types, déclarations des paramètres, déclarations des acteurs et exécution des événements.

### Déclarations des paramètres

La fécondité est paramétrisée par un risque de grossesse de base par intervalle d'âge de 2,5 ans en commençant au 15$^{e}$ anniversaire et par un facteur de risque relatif qui dépend de la situation d'union et de la durée. Nous définissons donc deux paramètres : AgeBaselinePreg1 et UnionStatusPreg1.

**Figure 4 : Paramètres de fécondité**




Pour les risques de fécondité, une partition du temps est utilisée pour définir les colonnes. Pour l'âge de référence, nous utilisons la partition AGEINT_STATE qui a été définie dans PersonCore.mpp. Pour définir les états d'union possibles pour les facteurs de risque relatif, le modèle utilise la classification UNION_STATE qui est déclarée dans PersonCore.mpp également.

```
parameters
{
	//EN Age baseline for first pregnancy
	double AgeBaselinePreg1[AGEINT_STATE];
	//EN Relative risks of union status on first pregnancy
	double UnionStatusPreg1[UNION_STATE];
};

parameter_group P02_Ferility		//EN Fertility
{
	AgeBaselinePreg1, UnionStatusPreg1
};
```

**Déclarations des acteurs**

Le seul état du module de fertilité est « parity_status », qui ne peut avoir que deux niveaux : « childless » et « pregnant ». (En effet, RiskPaths ne simule plus les événements de fécondité d'un acteur après la première grossesse.)

Dans Fertility.mpp, nous ne modélisons qu'un seul événement : la grossesse. La paire correspondante de fonctions d'événement est timeFirstPregEvent et FirstPregEvent.

```
classification PARITY_STATE                //EN Parity status
{
      PS_CHILDLESS,                                    //EN Childless
      PS_PREGNANT                                      //EN Pregnant
};

actor Person
{
      //EN Parity status derived from the state parity
      PARITY_STATE parity_status = {PS_CHILDLESS};

      //EN First pregnancy event
      event timeFirstPregEvent, FirstPregEvent;
};
```

**Exécution des événements**

Comme tous les événements Modgen, l'événement de première grossesse est exécuté en deux parties. La première détermine le moment de l'événement et la deuxième, les conséquences si l'événement a lieu. La fonction timeFirstPregEvent vérifie si l'acteur est couramment exposé au risque et, dans l'affirmative, tire une durée aléatoire en se basant sur le modèle de régression à risques constants proportionnels par morceaux paramétrisé par un âge de référence et un risque relatif selon la situation d'union. Par conséquent, le taux de risque est calculé d'après les deux paramètres AgeBaselinePreg1 et UnionStatusPreg1. Une durée aléatoire peut être obtenue à partir d'un nombre aléatoire suivant une loi uniforme par la transformation :

randdur=-log(RandUniform(1))/ taux de risque.

La fonction Modgen RandUniform() produit un nombre aléatoire à distribution uniforme compris entre 0 et 1. La fonction requiert un argument entier utilisé pour attribuer une chaîne de nombres aléatoires indépendants différente à chaque fonction de production d'un nombre aléatoire dans le code. S'il est omis, Modgen réécrit automatiquement un indice unique dans le fichier .mpp avant la conversion en code C++.

Lorsque l'événement a lieu, la valeur de l'état « parité » augmente de 1. (Il convient de souligner que l'état dérivé parity_status est placé automatiquement par « PS_PREGNANT ».)

```
TIME Person::timeFirstPregEvent()
{
      double dHazard = 0;
      TIME event_time = TIME_INFINITE;
      if (parity_status == PS_CHILDLESS)
      {

            dHazard = AgeBaselinePreg1[age_status]
              * UnionStatusPreg1[union_status];
          if (dHazard > 0)
          {
                event_time = WAIT(-log(RandUniform(1)) / dHazard);
            }
  }
  return event_time;
}

void Person::FirstPregEvent()
{
      parity_status = PS_PREGNANT;

}
```

### 4.3.3 Unions.mpp

La programmation des transitions entre les unions ne donne lieu qu'à l'introduction de nouveaux concepts mineurs dans la programmation Modgen, si bien que la discussion du code qui suit est limitée principalement à la dissolution des unions. Les taux de risque pour les premier et deuxième événements de dissolution d'une union sont sauvegardés dans le même tableau de paramètres parce qu'ils utilisent tous deux les mêmes intervalles temporels de durée d'union.

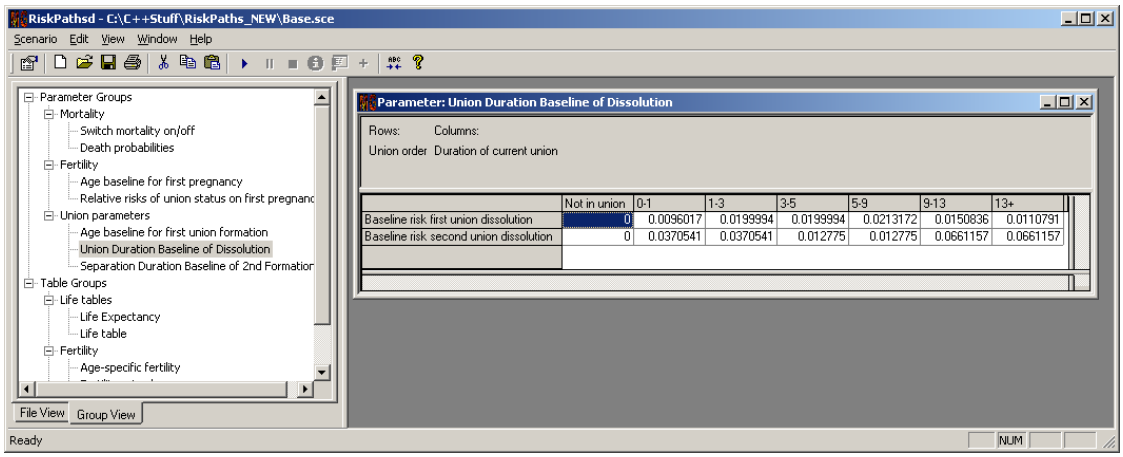
Afin de construire un paramètre au moyen des dimensions temps et ordre de l'union, nous définissons une partition du temps et une classification :

```
partition UNION_DURATION            //EN Duration of current union
{
      1, 3, 5, 9, 13
};

classification UNION_ORDER          //EN Union order
{
      UO_FIRST,          //EN First union
      UO_SECOND          //EN Second union
};

parameters
{
      …
      //EN Union Duration Baseline of Dissolution
      double UnionDurationBaseline[UNION_ORDER][UNION_DUR];
      …
};
```

**Figure 5 : Paramètres de dissolution d'une union**




Dans la fonction timeUnion1DissolutionEvent(), les taux de risque de dissolution de la première union sont obtenus de la façon suivante :

```
dHazard = UnionDurationBaseline[UO_FIRST][union_duration];
```

De la même façon, timeUnion2DissolutionEvent() fait référence à la deuxième ligne du paramètre :

```
dHazard = UnionDurationBaseline[UO_SECOND][union_duration];
```

Contrairement aux processus dont nous avons discuté jusqu'à présent, les processus de dissolution d'une union ne débutent pas à un moment prédéfini (p. ex. le 15$^{e}$ anniversaire), mais au moment des événements de formation d'une union. La durée d'une union est définie comme un état dérivé autoplanifié de la forme suivante :

```
//EN Currently in an union
logical in_union = (union_status == US_FIRST_UNION_PERIOD1
      || union_status == US_FIRST_UNION_PERIOD2
      || union_status == US_SECOND_UNION);

//EN Time interval since union formation
int   union_duration = self_scheduling_split(
            active_spell_duration( in_union, TRUE), UNION_DURATION);
```

En ce qui concerne la formation d'une union, l'exécution de l'horloge qui modifie l'état de durée de l'union union_status de US_FIRST_UNION_PERIOD1 à US_FIRST_UNION_PERIOD2 après trois ans dans une première union nécessite une certaine discussion. Contrairement aux états dérivés autoplanifiés utilisés pour toutes les autres horloges du modèle, ici — principalement pour illustrer cette alternative — nous exécutons

explicitement l'horloge comme un événement proprement dit. Cet événement a lieu après trois ans dans la première union. L'horloge est réglée au moment de la formation de la première union. La déclaration d'acteur comprend un état qui enregistre le moment du changement d'état, ainsi que la déclaration de l'événement.

```
actor Person
{
	…

	//EN Time of union period change
	TIME  union_period2_change = {TIME_INFINITE};

	//EN Union period change event
	event timeUnionPeriod2Event, UnionPeriod2Event;
};
```

Le temps pour le changement d'état est réglé dans l'événement de formation de la première union. Dans l'exemple de code, WAIT est une fonction Modgen intégrée qui produit le temps de l'événement courant plus une durée spécifiée (dans notre exemple, trois ans).

```
void Person::Union1FormationEvent()
{
	unions++;
	union_status = US_FIRST_UNION_PERIOD1;
	union_period2_change = WAIT(3);
}
```

L'exécution de l'événement est simple :

```
TIME Person::timeUnionPeriod2Event()
{
	return union_period2_change;
}

void Person::UnionPeriod2Event()
{
	if (union_status == US_FIRST_UNION_PERIOD1)
	{
		union_status = US_FIRST_UNION_PERIOD2;
	}
	union_period2_change = TIME_INFINITE;
}
```

### 4.4 Tables.mpp

Modgen fournit une fonction de totalisation croisée très puissante et souple pour communiquer les résultats du modèle. La programmation de chaque tableau de sortie ne requiert habituellement que quelques lignes de code. RiskPaths ne contient qu'un seul fichier de

tableaux qui regroupe les déclarations de tous ses tableaux de sortie. Cependant, pour des modèles plus détaillés, il est conseillé de regrouper les déclarations de tableau par groupes de comportement.

La syntaxe de base pour les tableaux est présentée à la figure 6. Les deux éléments centraux d'une déclaration de tableau sont les dimensions de classification de capture (qui définissent quand un acteur entre dans une cellule et quand il en sort) et la dimension d'analyse (qui enregistre ce qui se passe pendant que l'acteur est dans la cellule). Habituellement, les dimensions de classification sont des intervalles d'âge ou de temps (p. ex. fécondité par âge), des états (p. ex. fécondité selon la situation d'union) ou une combinaison des deux. Modgen ne limite pas le nombre de dimensions.

La dimension d'analyse peut contenir de nombreuses expressions, qui peuvent être des états ou des états dérivés. Modgen fournit une liste très utile de fonctions d'état dérivé spéciales qui enregistrent, par exemple, le nombre d'occurrences de certains événements, le nombre de changements dans les états, ou la durée de la présence de l'acteur dans les états. Deux concepts particulièrement utiles sont le mot-clé **unit** et la fonction d'état dérivé **duration()**. Le premier, c'est-à-dire **unit**, enregistre le nombre d'acteurs qui entrent dans une cellule d'un tableau, tandis que **duration()** enregistre le temps total qu'un acteur passe dans la cellule.

Les tableaux peuvent contenir des critères filtres pour définir si et dans quelles conditions les caractéristiques de l'acteurs seront enregistrées. Le meilleur moyen de comprendre les concepts de tableaux de Modgen consiste à examiner des exemples concrets tels que celui qui suit. Comme la richesse du langage de tableau de Modgen dépasse le cadre du présent chapitre, vous êtes invité à consulter le guide du développeur de Modgen.

**Figure 6 : Syntaxe des tableaux**

```
table actor_name table_name                    //EN table label
[filter_criteria]
{
      dimension_a *                            //EN dimension label
      …
      {
            analysis_dimension_expression_x,   //EN expression label
            …
      }
      * dimension_n                            //EN dimension label
      …
};
```

**Tableau 1 : Espérance de vie**

Le premier exemple de tableau contient les valeurs sommaires de notre simulation et ne possède aucune dimension, autrement dit les cellules s'appliquent à l'ensemble de la population sur l'entièreté de la période de simulation. Nous utilisons le mot-clé **unit** de Modgen, qui compte le nombre d'acteurs qui entrent dans une cellule d'un tableau (dans notre exemple, la simulation proprement dite) et la fonction **duration()** de Modgen qui additionne le temps que les acteurs passent dans cette cellule (dans notre exemple, le nombre total d'années vécues par tous les acteurs dans la simulation). L'âge moyen au décès de tous le acteurs dans la simulation est alors obtenu en divisant **duration()** par **unit**. En ce qui concerne les déclarations des paramètres, les commentaires placés dans le code sont utilisés comme étiquette dans l'application. (Notons que dans la déclaration de tableau qui suit, la partie « decimals=3 » du commentaire est utilisée pour déterminer le nombre de décimales dans les chiffres du tableau; cette partie du commentaire n'est pas transportée jusqu'à l'étiquette utilisée dans le rapport.)

```
table Person T01_LifeExpectancy //EN 1) Life Expectancy
{
 {
      unit,                    // EN Total simulated cases
      duration(),              // EN Total duration
      duration()/unit          // EN Life expectancy decimals=3
 }
};
```

**Tableau 2 : Table de mortalité**

Dans le deuxième tableau, nous enregistrons la population selon l'âge. Pour la sortie par âge, nous utilisons integer_age comme dimension du tableau.

```
table Person T02_TotalPopulationByYear    //EN Life table
{
      //EN Age
      integer_age *
      {
            unit,                         //EN Population start of year
            duration()                    //EN Average population in year
      }
};
```

**unit** et **duration()** font maintenant référence au nombre d'entrées et aux durées, respectivement, dans les intervalles d'âge d'un an. Donc, **unit** compte les acteurs présents au

début de chaque année, tandis que **duration()** fait référence à la population moyenne durant l'année.

### Tableaux 3 et 4 : Fécondité par âge

En plus du mot-clé **unit** et de la fonction d'état dérivé **duration()**, des états et un ensemble d'autres fonctions d'état dérivé peuvent être utilisés dans les tableaux. Si l'on utilise un état sans fonction, Modgen enregistre le changement de l'état pendant qu'il est dans une cellule particulière, c'est-à-dire la valeur de l'état au moment de la sortie de la cellule moins la valeur de l'état au moment de l'entrée dans la cellule.

L'expression parity/duration() enregistre la fécondité (par âge) comme étant le nombre d'événements de naissance divisé par le nombre moyen de femmes par année d'âge.

La deuxième expression est utilisée pour calculer le taux réel, c.-à-d. le nombre d'événements de naissance par durée de l'exposition. Une femme est exposée au risque d'une première grossesse quand elle est sans enfants. Nous divisons donc le nombre d'événements par le terme « duration( parity_status, PS_CHILDLESS ) ».

La dimension du tableau est l'âge en années complètes. Comme la fécondité est nulle jusqu'à l'âge de 15 ans et est très faible après 40 ans, les périodes d'âge avant 15 ans et après 40 ans ne sont pas subdivisées. Nous définissons donc une partition AGE_FERTILEYEARS qui est utilisée dans l'instruction self_scheduling_split qui définit la dimension du tableau.

```
partition AGE_FERTILEYEARS //EN Fertile age partition
{
      15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
      32, 33, 34, 35, 36, 37, 38, 39, 40
};

table Person T03_FertilityByAge     //EN Age-specific fertility
{
      //EN Age
      self_scheduling_split(age,AGE_FERTILEYEARS) *
      {
            //EN First birth rate all women decimals=4
            transitions(parity_status, PS_CHILDLESS, PS_PREGNANT) /
            duration() ,

            //EN First birth rate woman at risk decimals=4
            transitions(parity_status, PS_CHILDLESS, PS_PREGNANT) /
            duration( parity_status, PS_CHILDLESS )
      }
};
```

Le tableau 4 produit les taux de premières naissances pour les tranches d'âge de 2,5 ans utilisées pour la paramétrisation. Nous ajoutons également une autre dimension, à savoir la situation d'union; nous obtenons ainsi les valeurs simulées des paramètres du modèle.

```
table Person T04_FertilityRatesByAgeGroup //EN Fertility rates by age group
[parity_status == PS_CHILDLESS]
{
      {
            //EN Fertility decimals=4
            transitions(parity_status, PS_CHILDLESS, PS_PREGNANT) /
            duration()
      }
      * self_scheduling_split(age, AGEINT_STATE)      //EN Age interval
      * union_status                                  //EN Union Status
};
```

**Tableau 5 : Fécondité de la cohorte**

Le tableau 5 calcule deux mesures de fécondité de la cohorte — l'âge moyen à la première grossesse et l'absence d'enfants. Pour obtenir l'âge au moment de la grossesse, nous utilisons la fonction d'état dérivé value_at_transitions(parity_status, PS_CHILDLESS ,PS_PREGNANT, age) qui produit la valeur d'un état (âge) au moment d'une transition spécifique à un autre état, nommément quand parity_status change de PS_CHILDLESS à PS_PREGNANT.

```
table Person T05_CohortFertility          //EN Cohort fertility
{
      {
      //EN Av. age at 1st pregnancy decimals=2
      value_at_transitions(parity_status,PS_CHILDLESS,PS_PREGNANT,age)/
            transitions(parity_status, PS_CHILDLESS, PS_PREGNANT),

      //EN Childlessness decimals=4
      1 - transitions(parity_status, PS_CHILDLESS, PS_PREGNANT) / unit,

      //EN Percent one child decimals=4
            transitions(parity_status, PS_CHILDLESS, PS_PREGNANT) / unit

      }
};
```

**Tableau 6 : Grossesses selon la situation d'union et l'ordre de l'union**

Dans le tableau 6, nous utilisons un exemple de filtre qui déclenche la simulation d'une personne exactement à l'entrée dans un état, ici à la survenue de la grossesse. Nous nous

intéressons à la situation d'union à la première grossesse. Notons que ce filtre exclut aussi les femmes qui demeurent sans enfants.

```
table Person T06_BirthsByUnion //EN Pregnancies by union status & order
[trigger_entrances(parity_status, PS_PREGNANT)]
{
      {
            unit        //EN Number of pregnancies
      }
      *union_status+    //EN Union Status at pregnancy
};
```

**Tableau 7 : Risques de formation d'une première union**

Comme le tableau 4, ce tableau reproduit un tableau de paramètres. Bien qu'un tableau de sortie de ce genre ne contienne aucune information (si la taille d'échantillon est suffisamment grande, il devient très proche des paramètres originaux du modèle), il est utile pour valider le modèle et pour évaluer la variabilité Monte Carlo.

```
table Person T07_FirstUnionFormation      //EN First union formation
[parity_status == PS_CHILDLESS]
{
      //EN Age group
      self_scheduling_split(age, AGEINT_STATE) *
      {
            //EN First union formation risk decimals=4
            entrances(union_status, US_FIRST_UNION_PERIOD1)
            / duration(union_status, US_NEVER_IN_UNION)

      }
};
```

### 4.4.1 Groupement des tableaux de sortie

Comme les paramètres, les tableaux de sortie peuvent être groupés en vue de présenter les résultats de manière plus significative. Dans l'application RiskPaths, nous distinguons trois groupes de tableaux : les tableaux de mortalité, les tableaux de fécondité et les tableaux de situation d'union.

```
table_group TG01_Life_Tables        //EN Life tables
{
 T01_LifeExpectancy, T02_TotalPopulationByYear
};

table_group TG02_Birth_Tables       //EN Fertility
{
      T03_FertilityByAge, T04_FertilityRatesByAgeGroup, T05_CohortFertility
};
```

```
table_group TG03_Union_Tables       //EN Unions
{
      T06_BirthsByUnion, T07_FirstUnionFormation
};
```

### 4.5 Tracking.mpp

Le bloc de code track{} définit la liste des états qui doivent être enregistrés longitudinalement pour produire une sortie visuelle de BioBrowser. Cette commande est souvent placée dans des fichiers de tableaux. Dans notre modèle, cependant, nous avons décidé de coder un fichier Tracking.mpp distinct, puisque nous dépistons aussi les profils de risque calculés comme des états dérivés.

```
track Person
{
      integer_age,
      life_status,
      age_status,
      union_duration,
      dissolution_duration,
      unions,
      parity_status,
      union_status,
      preg_hazard,
      formation_hazard,
      dissolution_hazard
};
```

Le fichier comprend aussi la déclaration de trois états dérivés. Nous avons utilisé le concept d'état dérivé pour calculer les trois principaux taux de risque (grossesse, formation d'une union et dissolution d'une union) pour la sortie de BioBrowser. Nous le faisons à titre d'exemple uniquement, car tous les taux de risque, ventilés selon l'ordre de l'union, sont calculés dans les fonctions d'événement.

La déclaration des états dérivés « preg_hazard », « formation_hazard » et « dissolution_hazard » sont également de bons exemples de syntaxe pour construire des états dérivés à partir d'états simples à l'aide des conditions « if-else ».

```
actor Person
{
      //EN Pregnancy hazard
      double preg_hazard = (parity_status == PS_CHILDLESS) ?
            AgeBaselinePreg1[age_status] *
            UnionStatusPreg1[union_status] : 0;
```

```
      //EN Union formation hazard
      double formation_hazard = (union_status != US_NEVER_IN_UNION
            && union_status != US_AFTER_FIRST_UNION) ? 0 :
            ((union_status == US_NEVER_IN_UNION) ?
            AgeBaselineForm1[age_status] :
            SeparationDurationBaseline[dissolution_duration] );

      //EN Union dissolution hazard
      double dissolution_hazard = (union_status != US_FIRST_UNION_PERIOD1
&&
            union_status != US_FIRST_UNION_PERIOD2 &&
            union_status != US_SECOND_UNION) ? 0 :
            ((union_status == US_SECOND_UNION) ?
            UnionDurationBaseline[UO_SECOND][union_duration] :
            UnionDurationBaseline[UO_FIRST][union_duration]);
};
```

### 4.6 Fichier de traduction de langue RiskPathsFR.mpp

Ce fichier .mpp existe uniquement pour les modèles qui sont définis dans Modgen comme étant multilingues (ce qui, dans le cas de RiskPaths, signifie anglais et français). Toutefois, même dans un modèle bilingue, l'une des langues, l'anglais ou le français, est encore considérée comme étant la première langue ou langue primaire du modèle. L'anglais a été choisi comme langue primaire au moment où RiskPaths a été développé et, par conséquent, le fichier RiskPathsFR.mpp contient essentiellement les traductions des étiquettes et des notes du modèle dans l'autre langue, c'est-à-dire le français. (Si la langue primaire originale de RiskPaths avait été le français, ce fichier de traduction aurait été appelé RiskPathsEN.mpp et il aurait contenu les traductions en anglais des étiquettes et des notes du modèle.)

Normalement, toutes les notes et étiquettes sont entrées sous forme de commentaires de code dans les fichiers sources .mpp en utilisant la langue primaire du modèle, comme nous l'avons illustré à plusieurs occasions dans les exemples qui précèdent. Les traductions correspondantes sont subséquemment placées dans ce fichier .mpp distinct.

Un outil logiciel « Translation Assistant » d'appoint est disponible dans Modgen pour faciliter le processus de traduction en vérifiant l'ensemble complet de termes qui doivent être traduits et en ajoutant des versions uniques des entités traduites dans le fichier .mpp approprié (RiskPathsFR.mpp dans notre exemple).