HUGINEXPERT
The leading decision support tool

HUGIN EXPERT A/S
GASVÆRKSVEJ 5DK 9000
AALBORGDENMARK

WWW.HUGIN.COMCVR 14712984

HUGIN 不正検出マネジメント テクニカル・ホワイトペーパー

06/01/2009

この文書は、HUGIN 不正検出マネジメント（FDM）ソリューションに関する詳細なテクニカル情報です。HUGIN FDM の技術的および機能的特徴を説明し、HUGIN Java APを用いたHUGIN FDM の統合の詳細な実行例を提供します。

目次:

1 イントロダクション

このドキュメントの目的は、技術的および機能的な観点から、HUGIN 不正検出マネジメント (FDM)に基づく不正検出システムの開発と実装に関するタスクの概要を説明することです。 HUGIN FDM は、先端の統計モデルに基づく不正な（損害）保険請求を検出するための有効なソリューションです。このドキュメントは、HUGIN不正検出モデルを開発して、HUGIN Java APIを用いて既存のITプラットフォームに統合するプロセスを説明します。

図 1 は、（損害）保険請求の取り扱いプロセスでのHUGIN FDM システムの使用を説明しています。請求取扱い担当者は、請求通知書を受け取り、請求に関する情報を収集します。この情報は、請求取扱システムに入力され、その顧客について会社が持っているあらゆる情報と組み合わせられます。その請求を迅速な支払いに向かわせるか、請求調査員にその請求を渡すかの決定をサポートするために、ユーザー・インタフェースは簡単な“信号機“を備えています。“信号機“は、その請求が不正である尤度について、請求取扱い担当者が、よく情報に基づいて一貫性のある決定を行うための意思決定支援システムです。

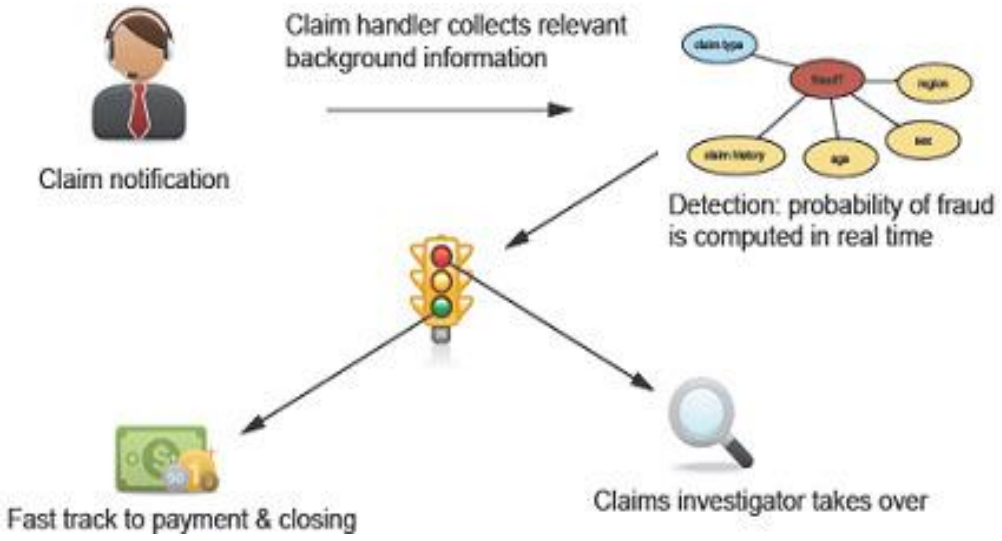


図 1 HUGIN FDM が請求の取り扱いプロセスでどのように使用されるかを説明する事例.

この事例では、“信号機” はHUGIN FDM システムによって制御されています。このドキュメントは、HUGINテクノロジーに基づくこのような信号機システムをどのように開発するかを説明します。

このドキュメントの対象読者は、モデル開発に携わる保険計理士とシステム・インテグレーション・プロセスに携わるITスタッフ・メンバーです。

2 HUGIN 不正モデルの事例

モデル・ベースの意思決定支援システムの中核コンポーネントは、モデルです。不正検出モデルは、不正請求と顧客の振る舞いや顧客の特徴を含む数々の不正指標やリスク特徴の間の従属性を特定します。とくに、不正検出モデルは、不正請求と正当請求の間で、どの振る舞いや顧客の特徴が異なるかを特定します。

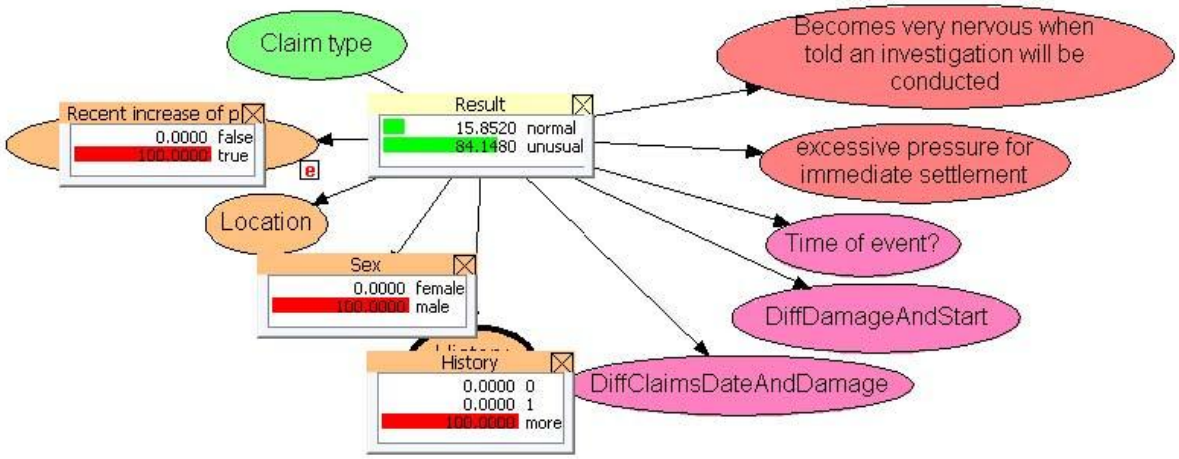
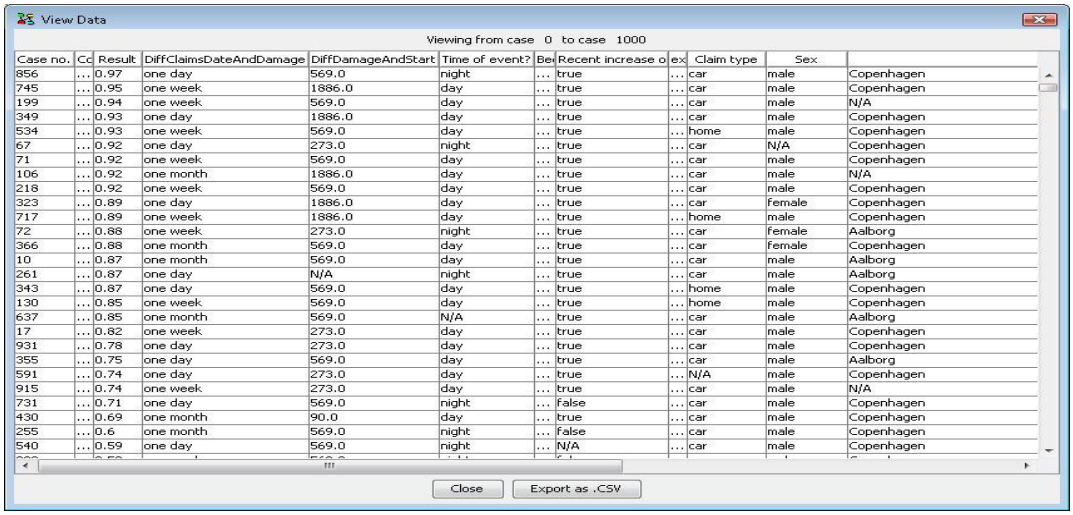


図 2 簡単なHUGIN 不正検出モデル.

図2は、簡単な不正検出モデルを示します。このモデルは、請求の種類、不正、請求履歴、年齢、などの間の従属関係を記述しています。複数回の請求履歴のある男性が、最近、保険範囲を増加させて盗難請求を行ったケースでは、このモデルによると、その請求が不正である確率は84.15%です。限定された情報であっても、モデルは不正の確率を評価できます。このケースでは、不正の高い確率を示す情報が得られました。

HUGIN FDM ソリューションは、図3に示すように、不正の確率によって並べ替えされた請求のリストを生成するために使用することもできます。このリストは、“Result(結果)”と名付けられた列の値によって降順に並べられています。



Case no.	Cc	Result	DiffClaimsDateAndDamage	DiffDamageAndStart	Time of event?	Be	Recent increase o	ex	Claim type	Sex	
856	...	0.97	one day	569.0	night	...	true	...	car	male	Copenhagen
745	...	0.95	one week	1886.0	day	...	true	...	car	male	Copenhagen
199	...	0.94	one week	569.0	day	...	true	...	car	male	N/A
349	...	0.93	one day	1886.0	day	...	true	...	car	male	Copenhagen
534	...	0.93	one week	569.0	day	...	true	...	home	male	Copenhagen
67	...	0.92	one day	273.0	night	...	true	...	car	N/A	Copenhagen
71	...	0.92	one week	569.0	day	...	true	...	car	male	Copenhagen
106	...	0.92	one month	1886.0	day	...	true	...	car	male	N/A
218	...	0.92	one week	569.0	day	...	true	...	car	male	Copenhagen
323	...	0.89	one day	1886.0	day	...	true	...	car	female	Copenhagen
717	...	0.89	one week	1886.0	day	...	true	...	home	male	Copenhagen
72	...	0.88	one week	273.0	night	...	true	...	car	female	Aalborg
366	...	0.88	one month	569.0	day	...	true	...	car	female	Copenhagen
10	...	0.87	one month	569.0	day	...	true	...	car	male	Aalborg
261	...	0.87	one day	N/A	night	...	true	...	car	male	Aalborg
343	...	0.87	one day	569.0	day	...	true	...	home	male	Copenhagen
130	...	0.85	one week	569.0	day	...	true	...	home	male	Copenhagen
637	...	0.85	one month	569.0	N/A	...	true	...	car	male	Aalborg
17	...	0.82	one week	273.0	day	...	true	...	car	male	Copenhagen
931	...	0.78	one day	273.0	day	...	true	...	car	male	Copenhagen
355	...	0.75	one day	569.0	day	...	true	...	car	male	Aalborg
591	...	0.74	one day	273.0	day	...	true	...	N/A	male	Copenhagen
915	...	0.74	one week	273.0	day	...	true	...	car	male	N/A
731	...	0.71	one day	569.0	night	...	false	...	car	male	Copenhagen
430	...	0.69	one month	90.0	day	...	true	...	car	male	Copenhagen
255	...	0.6	one month	569.0	night	...	false	...	car	male	Copenhagen
540	...	0.59	one day	569.0	night	...	N/A	...	car	male	Copenhagen

図 3 不正の確率によって並べ替えられた請求リスト

図 3 は、HUGIN Graphical User Interfaceを用いて生成されたリストの一例うぃ示します。HUGIN アプリケーション・プログラミング・インタフェース (API)を用いて、同様な機能を既存のITプラットフォームに観点に簡単に組み込めます。

3 HUGIN 不正モデルの構築

HUGIN FDM ソリューションの中核コンポーネントは、上記に述べたように、その請求と顧客について与えられた情報に基づいて、不正の確率を計算するために使用するモデルです。原則的に、そのモデルはいくつでも変数を含むことができ、どのようなグラフ構造を持つこともできます。図2に示すような比較的少ない数の指標と単純な構造のモデルが、目覚ましい性能を持つ可能性があることが経験的に知られています。

HUGIN 不正モデルは、専門家のドメイン経験・知識、または過去データ、またはこれらの2つのソースの組み合わせにから構築されます。不正モデルの典型的な構築プロセスは、下記のステップからなります:

1 データ収集と準備。
2 ネットワーク構造の指定。
3 専門家の知識の指定とパラメータ推定。
4 モデル評価。

HUGIN FDM システムは、不正モデル中で表現された知識とモデル中で伝播された不正に関する情報に基づいて、請求が不正である確率を計算します。

不正モデル中の各ノードは、*目的*変数（仮説またはクラス分類変数ともいう）、すなわち、請求が不正であるか、または疑わしいかを示す変数、または、*指標*変数、たとえば、不正と非不正の保険請求を区別するために用いられる請求や顧客の特徴を表します。不正モデルは、目的ノードでも指標ノードでもないノードも含む場合もあります。また、不正モデルは、図2に示す単純な構造だけでなく、どのような非巡回グラフ構造、有向グラフ構造もとることができます。図2の構造は、不正モデルの最もよくある構造で、このタイプのモデルが不正予測によく適していることが経験的にわかっています。

図 4 は、HUGIN Graphical User Interfaceにおける図2のモデルを示して、変数 “Sex”、“Recent increase of policy limits” および“DiffClaimsDateAndDamage”が開かれています。各表は、変数“Result”を仮定した（与えた）ときの変数の条件付き確率を表しています。各表のエントリーは、ドメインの専門家の知識と経験から評価されるか、過去データから推定されるか、またはこの2つの組み合わせです。たとえば、図4のに示された“Sex”に関する表の左上隅の0.3の数字は、“Result=normal”を仮定した“Sex=female”の条件付き確率です。この数字は、専門家の知識と経験で評価されるか、または、データから推定されます。

モデルは、保険会社が保険不正に関して持っている知識、経験および、過去データの定量的表現です。この知識、経験およびデータは、会社が運営されてシステムが使用される限り、増え続けます。会社は、システムの使用経験を徐々に構築します。この新しい知識、経験およびデータが、徐々にモデルに統合されます。システムの更新を確かにするために、我々は、少なくとも毎年1回はモデルを再考することを薦めます。これは、母集団での変化が知識表現に組み込まれることを確かにします。

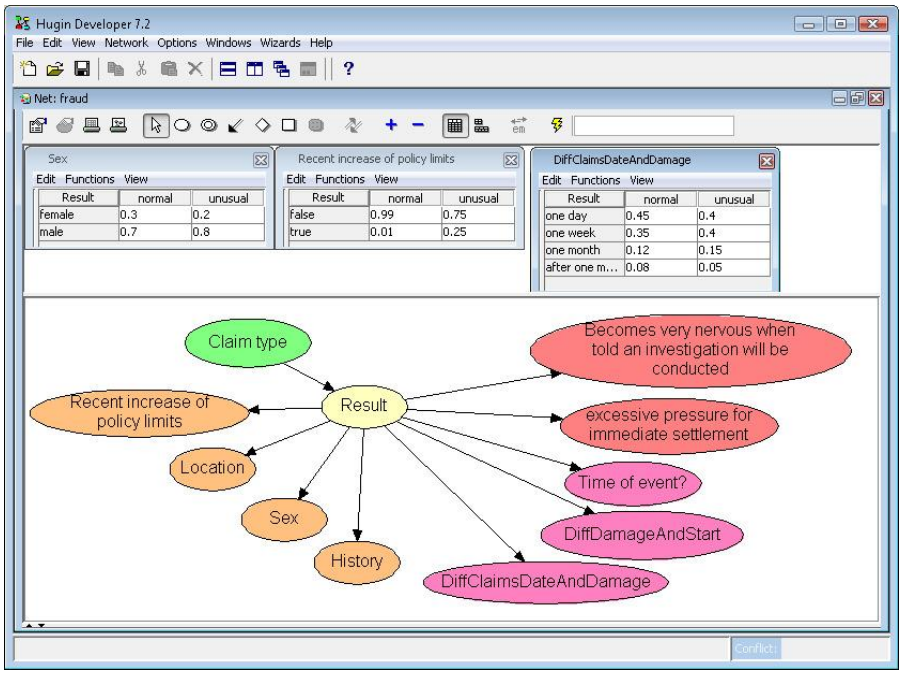


図 4　HUGIN FDM モデル.

次の節は、HUGIN FDM システムを既存のIT プラットフォームに統合する方法を説明します。 2 つの主要なタスクは、 1) 不正モデル内の各変数を適切なデータソースにリンクすること、そして2) 更新された確信度が計算されると、不正の確率を読んで表示すること、です。

統合の事例

HUGIN FDM ソリューションの既存ITプラットフォームへの統合は、HUGIN APIを用いて実行する簡単なタスクです。統合を実行する前に、不正の確率がユーザーにどのように表示されるべきかという重要な決定をしなければなりません。我々は、赤、黄、緑（または赤と緑のみ）の“信号機”を用いて不正の確率を表示することを推奨します。

統合は 3 つのステップからなります:

1　各指標変数を、既存のITプラットフォームのデータソースにリンクさせる。
2　目的変数での不正に反映されるステートの確信度を、 “信号機” （この表示が使用される場合）にリンクさせる。
3　関連するデータを収集して入力するための方法を実装し、不正モデルを用いて確信度を更新し、そして、結果を表示する。

付録 A は、図２に示すモデルを用いて統合がどのように実行されるかを説明する完全な事例です。この事例は、HUGIN Java API を用い、下記のステップからなります:

1　ファイルから不正モデルを読み込む。
2　モデル中に表現された変数を把握する。
3　モデルを計算用の構造にコンパイルする。
4　事例請求のオブザベーションを入力する。
5　請求が不正である確率を計算するためにエビデンスを伝播する。
6　結果を表示する。
7　請求に関連するオブザベーションを消去する。

ステップ 4 から 7 は、請求の不正確率が計算されるたびに繰り返されます。あとのステップは、1回だけ実行されます。

HUGIN FDM ソリューションの主な強みの 1 つは、ナレッジベース、すなわち、不正モデルと、既存ITプラットフォームへの不正モデルの統合の明確な分離です。図 5 は、モデル開発（通常、HUGIN のコンサルタントがクライアント企業様で分析を行います）とシステム統合（通常、HUGIN のコンサルタントのアシストによりクライアント企業様内のITスタッフが実行します）の間の明確な分離を説明しています。

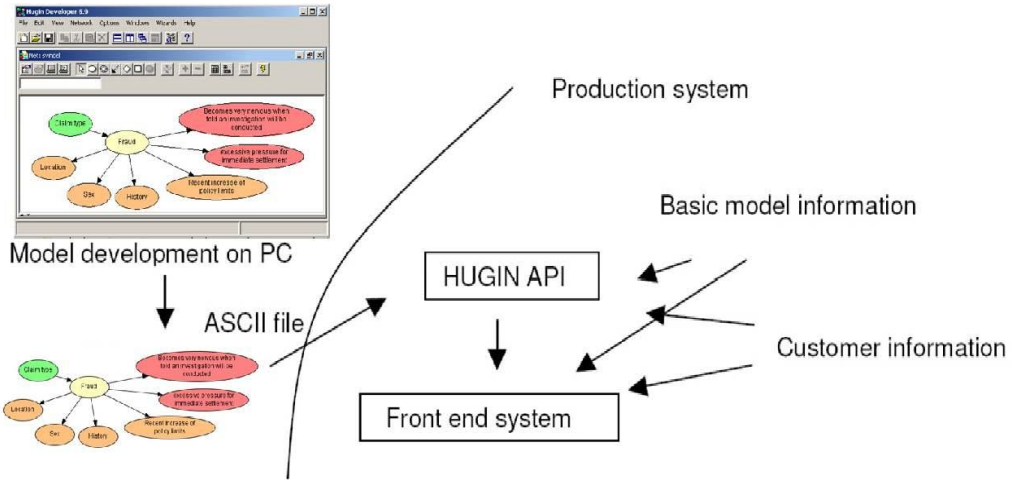


図 5 HUGIN FDM は既存プラットフォームと統合して、モデル開発とシステム統合の間の明確な分離を提供する。

統合が完了すると、既存のモデルを新しいモデルに置き換えるのは簡単な作業です。モデルに新しい変数が追加されていない場合、それは古いモデルの代わりに新しいモデルを読み込むだけのことです。モデルに新しい変数が追加されている場合、その変数が適切なデータソースにリンクされなければなりません。それがこれです。

HUGIN APIを用いてアプリケーションをコンパイルして実行する方法についての情報も含めて、さまざまなテクニカル・プラットフォームでのHUGIN APIの使用のさらに技術的詳細については、HUGIN API リファレンス・マニュアルの 1 章を参照してください。

テクノロジー

HUGIN FDM で使用されているテクノロジーは、ベイジアンネットワーク（または、ベイジアン信念ネットワーク、因果確率モデル、ベイズネットなど）として知られています。このテクノロジーとHUGIN ソフトウェアは、競合システムに対して多くの利点を持ちます:

- 専門家の知識と過去データの組み合わせ。
- データ中の欠損値の取扱い、不完全データでの計算が可能。
- 推論エンジンがとても効率的で、リアルタイムの推論が可能。
- モデルと実装がフレキシブルで、保守、拡張、修正が簡単。
- モデルを既存のシステムに統合するのが簡単。
- モデルのグラフィカルな性質のお陰で、モデルを伝達するのが簡単。

さらに、このテクノロジーとHUGIN ソフトウェアは、情報の価値分析、パラメータ感度分析、エビデンス感度分析、および矛盾分析をサポートします。これらの分析手法は、不正モデルの構築とシステム活用の両方で役立ちます。

6 機能情報

この節は、最も便利な情報と簡単な参照を提供するために質問と回答で構成されています。

6.1 どの保険セグメント用の不正検出モジュールがあるか?

HUGIN 不正検出モデルは、請求取扱い担当者やSIUエージェントのようなクライアント・ドメインの専門家とHUGINのコンサルタントの間での協力で開発されます。モデルは、ドメイン専門家や過去データへのアクセスを前提として、すべてのセグメント（すなわち、商品タイプ）について開発できます。一般的なアプローチは、1つのセグメントについてモデルを開発し、他のセグメントのモデルを開発する開始点として、そのモデルを使用するというものです。このテクノロジーとツールは、モデルが継続的に改善され拡張されるようなインタラクティブな開発プロセスをサポートしていて、とてもフレキシブルです。

6.2 請求取扱いプロセスでの不正検出オプションは?

HUGIN **不正モデルは、**"不正"変数と性別、年齢、請求履歴などの指標変数の間の確率的従属関係を記述します。モデルは、与えられた請求についてリアルタイムで不正の確率を計算します。モデルは不完全な情報（欠損値）で計算します。したがって、新しい情報が明らかになるたびに、不正の確率をリアルタイムで計算することができ、ツールは、モデル中で表現されている適切な質問を示唆することができます。処理時間はモデルの複雑さによりますが、最も一般的な不正モデルでは、時間と空間の複雑さは、指標変数の数に対して線形的です。不正検出モデルは、確率的グラフィカル・モデル（ベイジアンネットワーク）で、通常、静的で、年に数回の更新で済みます。HUGIN Graphical User Interfaceは、手動の修正と、データと専門家知識からの再トレーニングの両方の機能を備えています。モデルが既存のITプラットフォームに統合されると、モデルが新しい変数で拡張されたときだけ、追加の統合作業が必要になります。モデル開発とモデル統合の間には、明確な区分があります。

6.3 請求取扱いプロセスの追加ワークフローのサポートは？

HUGIN ツールは、"予期しない" オブザベーションを決定する矛盾分析、最も"価値のある"次のオブザベーション、たとえばクライアントへの質問、を決定する情報の価値をサポートし、モデルの複雑さに基づいて処理時間が計算できます。

6.4 どのような分析能力 / 調査のサポートがあるか？

HUGIN Graphical User Interfaceは、過去データから、変数間の相関を識別したり、変数間の相関の強さを推定したりする、さまざまなデータ解析能力を持ちます。矛盾分析機能は、オブザベーションの"予期しない" 集合の識別をサポートします。このツールは、感度分析、情報の価値分析、矛盾分析をサポートします。

7 技術情報

この節は、最も便利な情報と簡単な参照を提供するために質問と回答で構成されています。

7.1 ソフトウェアの構造は?

HUGIN ソフトウェア・パッケージは、不正モデル開発のためのHUGIN Graphical User Interfaceと、不正モデル統合のためのHUGIN Decision Engine からなります。HUGIN Decision Engine は、C、C#、C++、Java および VBAのAPIを持ちます。HUGIN Graphical User Interfaceは、（JNIを用いた）Java 実装であり、HUGIN Decision EngineはISO Cを用いて実装されています。

7.2 データ要件は?

HUGIN Graphical User Interfaceは、データベース・アクセスのためにODBC/JDBCインタフェーをス持ち、CSV

ファイルを読むこともできます。HUGIN Decision Engineは、リアルタイムのデータ入力、ファイルからの読み込みのためのAPI機能を持ちます。HUGINツールは、どのようなデータもストアしません。

7.3 既存のワークフローにツールを統合する方法は?

不正検出モデルは、HUGIN Decision EngineのAPIの1つを用いて、既存のシステムに統合されます。APIは、C、C#、C++、Java および VBA用があります。HUGIN Decision Engine は、情報を入力して、不正の確率を計算し、不正の確率にアクセスするために必要な機能を持ちます。不正検出モデルは、通常、請求取扱い担当者の既存のユーザー・インタフェースに信号機として実装されます。

7.4 必要なインタフェースの技術要件は?

HUGIN Graphical User Interfaceは、HUGIN Decision Engine用のJava API 上にJavaで実装されます。HUGIN Decision Engineのコアは、ポータビリティと効率の理由から、ISO Cを用いて実装されています。HUGIN Decision Engine は、Windows やLinuxが動作しているPC、Mac OSが動作しているMac、Solarisが動作しているPCやSUN サーバー、HP-UX が動作しているHPサーバー、そして IBM メインフレームなど、幅広いハードウェア・プラットフォームおよびオペレーティング・システムに展開されてきました。

7.5 ツールの性能は?

HUGIN Decision Engine は、ユーザー・アドミニストレーションをサポートしません。モデルが、たとえば、同時に複数のユーザーからアクセスされる場合、HUGIN Decision Engineのユーザーは、相互排除変数の使用に関与します。代案は、同じモデルを実行するプロセスのダイナミック・プールを持つ場合があります。不正検出モデルの最も一般的なタイプでの時間と空間の複雑さは、指標変数の数に対して線形的です。反応時間は、ハードウェア次第です。

7.6 ツールのアドミニ要件は?

HUGIN Graphical User Interfaceは、ユーザー、すなわち不正検出モデルを開発しているアナリストのデスクトップ・コンピュータにインストールされます。HUGIN Decision Engine は、ホスト・コンピュータ上にインストールするために、単一のファイルまたは一対のファイルとして配布されます。ユーザー・アドミニストレーションはありません。HUGIN 不正検出モデルは、単一のファイル（通常ASCIIファイルとして）に格納されます。このファイルは、HUGIN Graphical User InterfaceとHUGIN Decision Engineを用いてアクセスできます。既存のモデルにマイナーな修正を行った場合、更新は、サーバー上のこのファイルを置き換えるのみです。

7.7 ソフトウェアの更新は?

HUGIN Graphical User Interface は、分離された1つのユーザー・プログラムです。HUGIN Graphical User Interface は、HUGIN のwebサイトにリリースされた新しいビルドを自動的にインストールする機能を持ちます。HUGIN Decision Engineの新しいバージョンは、ホスト上の既存のファイルと置き換えることでインストールされます。ソフトウェア・パッケージの新しいバージョンは、年に1-2 回リリースされます。HUGIN Decision Engine の機能は、テスティング・プログラムを用いてテストされます。更新は強制的ではありません。新しい更新は、通常、新しい機能が必要なときに、ホスト上のみにインストールされます。これは、不正検出モデルでは稀です。

付録 A. Java **事例ソースコード**

```
/*
        * この簡単な Java プログラムは、簡単なHUGIN 不正検出モデルの統合を説明します。
        * 目的は、特定の保険請求について不正の確率を計算することです。
        * 請求の情報がモデルに入力され、不正の事後確信度が計算され、標準の出力に表示されます。
        *
*

        * アプリケーションはこのコマンドでコンパイルされます:
        * javac -cp hapi71.jar;. FraudIntegration.java
        * ここで hapi71.jar は HUGIN Java API (version 7.1)です。
*

        * このアプリケーションは、このコマンドで実行されます (HUGIN java API に関連づけられた
        * dll/so ファイルが、現在のディレクトリに置き換えられます):
        * java -cp hapi71.jar;. FraudIntegration fraud.net ここで
        * fraud.net は不正モデルが定義された HUGIN ネットワークです。
*

        * 標準での出力は次のようになります: P(Result=unusual | evidence) = 76,05%
*

        * 著作者: Anders L Madsen @ HUGIN EXPERT A/S
*

        * For any questions or comments, please contact the author at: alm@hugin.com

 */ import COM.hugin.HAPI.*; import java.text.DecimalFormat;

class FraudIntegration { // the
   model Domain dom = null;

   // the nodes / variables in the modelLabelledDCNode Result, Sex, ClaimType;// ...

   // NumberedDCNode ...; // none in model

   IntervalDCNode DiffDamageAndStart;// ...

   BooleanDCNode PolicyIncrease;// ...

   // ContinuousChanceNode ...; // none in model

   public FraudIntegration (String name)
   {

         try {// load model from file. Done once.dom = new Domain (name, new DefaultClassParseListener ());

            // Get handles for the nodes of the model. One list of// nodes for each node type. Done once.
```


```
            // ContinuousChanceNodes ... none in the example

            // LabelledDCNodes Result = (LabelledDCNode)dom.getNodeByName("Result"); Sex =
            (LabelledDCNode)dom.getNodeByName("sex"); // Notice that node name is unique, not
node label ClaimType = (LabelledDCNode)dom.getNodeByName("type"); // ...

            // NumberedDCNodes// ...

            // BooleanDCNodesPolicyIncrease = (BooleanDCNode)dom.getNodeByName("increase");// ...

            // IntervalDCNodesDiffDamageAndStart =
            (IntervalDCNode)dom.getNodeByName("DiffDamageAndStart");// ...

        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}


// Main procedure. This is where the actual calculations are performed.
protected void doCalculations () {DecimalFormat d = new DecimalFormat ();d.setMaximumFractionDigits (2);

        try{// compile the model for inference / analysis. Done once.dom.compile ();

            // link nodes (indicators) of model to data sources, use// node name as unique identifier. In the example
            data is// hard coded into the source code. The link between the// data source and HUGIN is made once.

            // in principle we should now have a loop over all cases,// but we only consider a single case// 1) insert
            information from caseinsertEvidence ();

            // 2) propagate evidencedom.propagate(Domain.H_EQUILIBRIUM_SUM,
            Domain.H_EVIDENCE_MODE_NORMAL);

            // 3) show belief of Result for the case

            System.out.println ("P(Result="+Result.getStateLabel(1) +" | evidence) = " + d.format(Result.getBelief (1)
                                    * 100) +"%");
```


```
            // 4) remove observations to prepare for next casedom.initialize ();

            // do steps 1-4 for all cases or each time the probability// of fraud is to be updated (evidence can be
            entered// incrementally. A propagation is required prior to// accessing the posterior belief of Result
          } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
          }
}


// It is very important that the data values entered matches the// state labels/values. Otherwise they are ignored.
State labels// are case sensitive. Some values may need to be computed if not// presented in the data source
(e.g., age)////protected void insertEvidence () {
    try{ // case: auto claim of male with recent increase in policy
        and claim made 186 days after policy starts

            enterObservation (ClaimType, "car");enterObservation (PolicyIncrease, "true");enterObservation
            (DiffDamageAndStart, 186);enterObservation (Sex, "male");

            //... more data
          } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
          }
}


      // There is one enterObservation method for each
      node typeprotected void enterObservation
      (ContinuousChanceNode n, double value)
      {try{n.enterValue (value);
        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}

      // Notice that select state is case sensitive. getStateIndex
      returns -1 for unmathed values.protected void
      enterObservation (LabelledDCNode n, String label)
      {try{n.selectState (n.getStateIndex (label));
```


```
        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}


         protected void enterObservation
                (BooleanDCNode n, String label)
                {try{n.selectState
                (n.getStateIndex (label));
        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}


         protected void enterObservation
                (NumberedDCNode n, double
                value) {try{n.selectState
                (n.getStateIndex (value));
        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}


         protected void enterObservation
                (IntervalDCNode n, double value)
                {try{n.selectState
                (n.getStateIndex (value));
        } catch (Exception e) {e.printStackTrace ();System.err.println (e.getMessage ());
        }
}


static public void main (String args[])
{

        if (args.length != 1) {System.out.println ("usage: <net>");System.exit (-1);
        }FraudIntegration fi = new FraudIntegration (args[0]);fi.doCalculations ();
} }
```

付録 B. システム・パフォーマンス

我々は2つの異なる不正検出モデルの事例でHUGIN FDM ソリューションの性能を評価してデモを行う実験を行いました。モデルMは、HUGIN FDM の顧客の1社で使用された18個の指標によるモデルで、モデルN は、9個の指標による同じモデルです。

HUGIN FDMの性能を評価するために、異なるサイズのデータ集合のモデルのパラメータを推定し、同じデータ集合内の各ケースの不正の確率を計算する使用時間を分析しました。

実験の結果が、下記の表に示されます。時間は秒で指定されています。実験は、HUGIN Java API version 7.2 と Java 6 update 17 を用いて、デスクトップ・コンピュータ (CPU i7 920, 12 GB RAM, Windows Vista)で実行されました。

パラメータ推定の性能が、下の表に示されています(繰り返し数のしきい値は 0 で、収束しきい値は 0.0001)。

	データサイズ			
モデル	10000	50000	100000	500000
M (18)	1.08	5.39	10.89	54.45
N (9)	0.23	0.77	1.53	7.66

表 1: パラメータ推定のランタイム・パフォーマンス（単位は秒）。

確率計算（確信度の更新）の性能が、下の表に示されています。これは各ケースをエビデンスとしてモデルに入力すること、エビデンスの伝播、周辺確信度の計算を含みます。

	データサイズ			
モデル	10000	50000	100000	500000
M (18)	0.74	3.80	7.13	34.91
N (9)	0.37	1.83	3.65	18.15

表 2: 確信度更新のランタイム・パフォーマンス（単位は秒）。

データ集合は、5% の欠損値（完全にランダムで欠損）で本当のモデルからランダムに生成されました。実験は、標準的なデスクトップPCでのHUGIN FDM のハイパフォーマンスを示します。使用時間は、各モデルのデータ集合のサイズによって、ほぼ線形的に増大します。

結論は、このタイプのモデルのパラメータ推定はとても効率的で、大規模なケースの集合がバッチで分析できる、ということです。バッチとリアルタイムの確信度更新の両方ともが、この実験で使用されたタイプのモデルでとても効率的です。たとえば、モデル M では、確信度更新の性能は、ほぼ14.000ケース/秒であるのに対して、モデルNの性能は、モデル Mの性能のほぼ2倍です。

付録 C. その他の情報

図6 は、HUGIN APIのアーキテクチャを説明しています。HUGIN Decision Engine は、ISO Cで実装されています。これはHUGIN Decision Engine を高度にポータブルかつ効率的にします。APIのISO C 実装の最上位レイヤが実装されています。APIは、C、 C++、 C#、 Java およびアプリケーション用のVisual BasicのActiveX Server などの主要なプログラミング言語向けに実装されています。

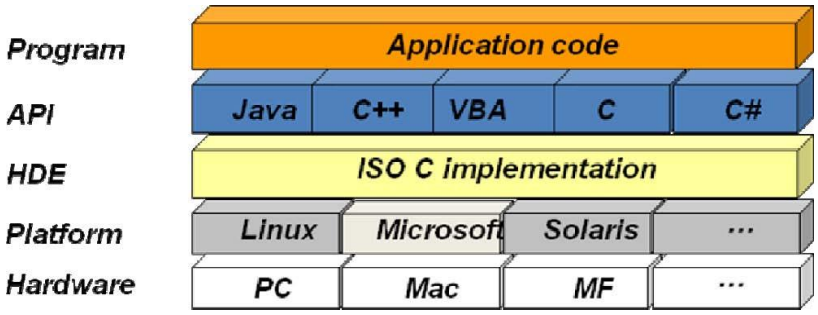


図 6 HUGIN API アーキテクチャ

HUGIN Decision Engine はISO Cで実装されているので、とてもポータブルです。HUGIN Decision Engine のインスタンスは、Linux、Microsoft Windows、 SUN Solarisなどを含むたくさんの異なるプラットフォームで実行できることが知られており、PDAから MacやPC、そしてメインフレーム・システムに至るまでのさまざまなハードウェア・プラットフォームで実行されてきました。HUGIN Decision Engine は、マルチ・ユーザーおよびマルチCPU プラットフォームで実行でき、32ビットと64ビットの両方のバージョンがあります。

HUGIN API リファレンス・マニュアルの第1章は、さまざまなプラットフォームやハードウェアでのHUGIN APIの使用についての詳細を説明しています。