pure::variants デモ概説書

目次

1. はじめに

この資料では、簡単な気象監視システムを例題として pure::variants によるモデリング環境とバリアビリティー/バリアント管理機能の使い方を説明しています。

pure::variants を用いて製品ファミリーのバリアントとバリアビリティーを分類し管理することで、体系的に資産を再利用して効率良く開発し、メンテナンスできるようになります。

弊社 pure::variants 製品情報サイト「プロダクトライン開発のバリアント管理」にある「各種資料と動画デモ」をご覧いただくことで、効率的に評価を行っていただけると存じます。また、この例題の説明（本資料）とデモプロジェクトは、同サイト内の「ソースコードやフラグのバリアント管理」「サンプルプロジェクト」にあります。

2. 準備

① pure::variants の評価版や正式版を用意してインストールする

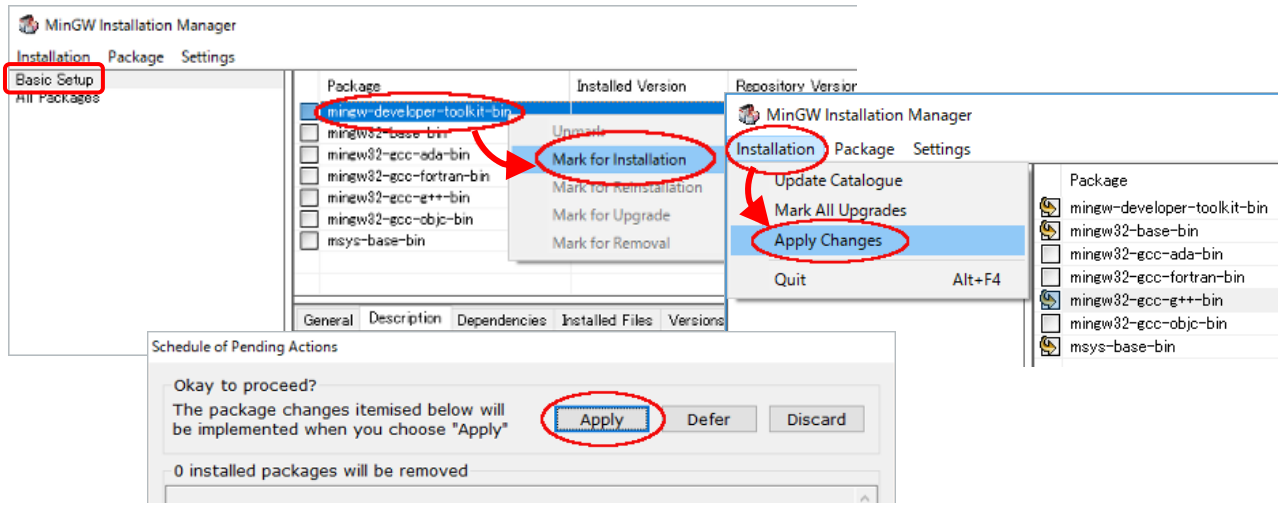
② サンプルプロジェクトをコピーする

purevariants_demo.zipを解凍し、`WebinarFinished` ディレクトリを〈ワークスペース〉にコピーします。（pure::variants バージョン6の場合 〈ワークスペース〉は、デフォルトのインストールでは `C:¥Users¥〈ユーザー名〉¥pure-variants-workspace-6.0` となっています）

このzipファイルは https://www.fuji-setsu.co.jp/files/purevariants_demo.zip からダウンロードして下さい。

③ ビルド環境をインストールする

C++コンパイラとビルド環境がなければ、Windows 上で実行可能なフリーの GNU コンパイラ・ツールチェーンである MinGW とその実行のためのシェル環境 MSYS をインストールください。MinGW プロジェクト日本語トップページ（https://ja.osdn.net/projects/mingw/）から mingw-get-setup.exe [1] をダウンロードして実行すると、MinGW Installation Manager 画面になります。Basic Setup で mingw-developer-toolkit-bin、 mingw32-base-bin、 mingw32-gcc-g++-bin、 msys-base-bin を選択し、メニューから Installation > Apply Changes を選択し、Apply をクリックしてインストールします。[2]



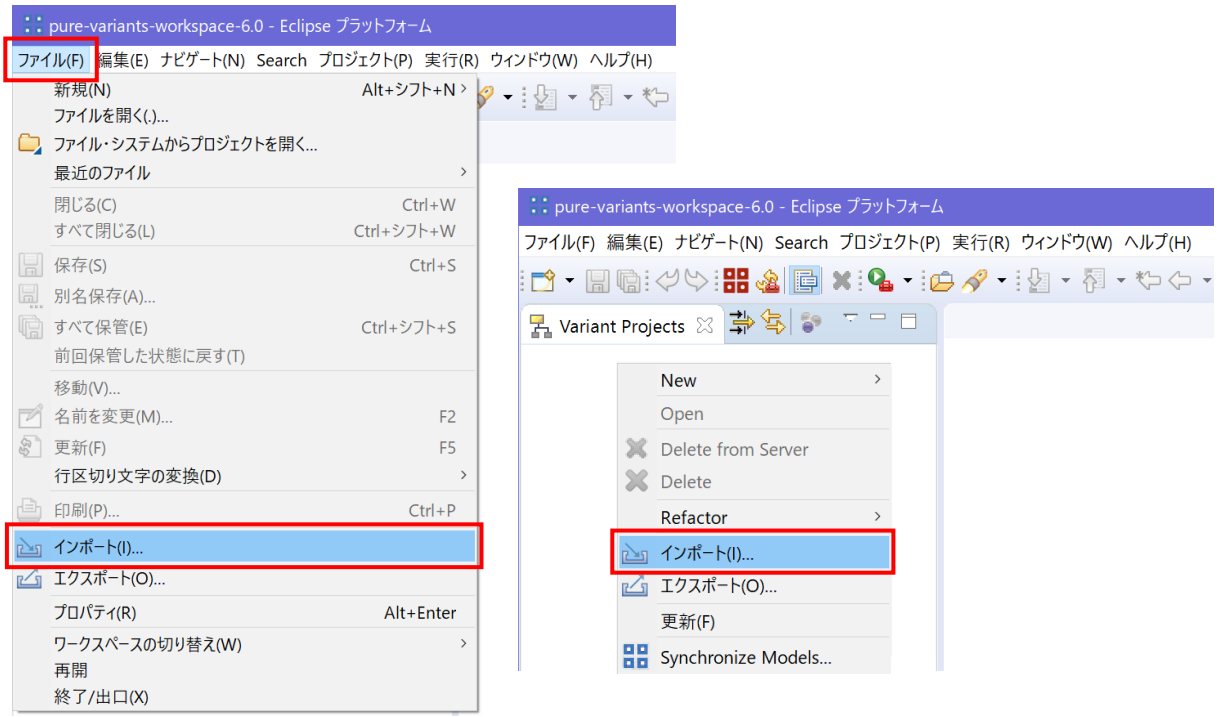
[1] 現在、2017-09-06 版です。

[2] MSYS の代りに MSYS2 や Cygwin、WSL にインストールされた Linux などを使用することもできます。

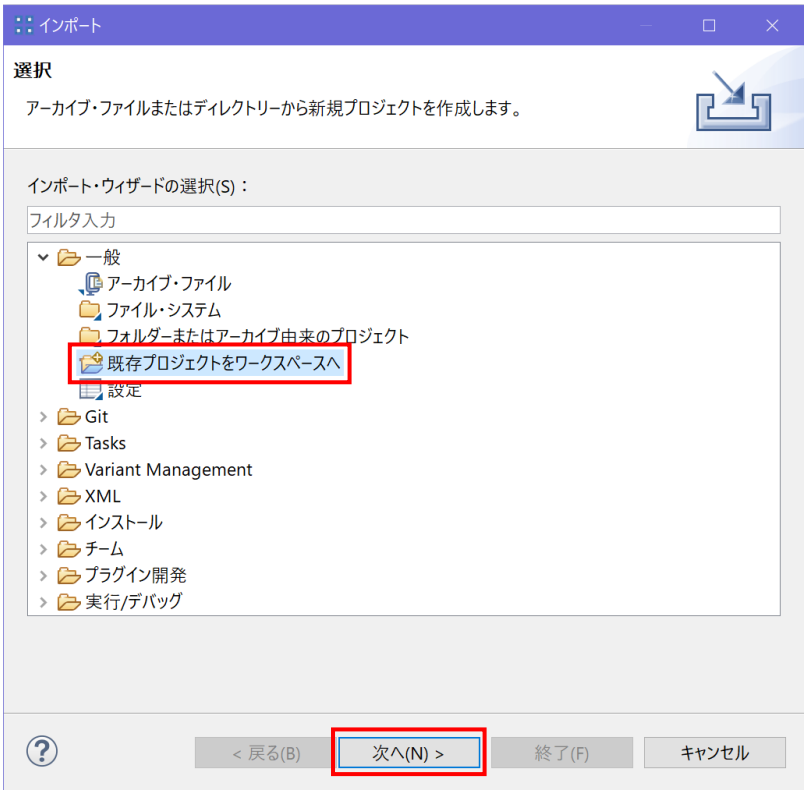
3. プロジェクトの作成

通常、プロジェクトは新規に作成しますが、ここでは既存のプロジェクトをインポートする方法をとります。以下の手順でプロジェクトをインポートできます。

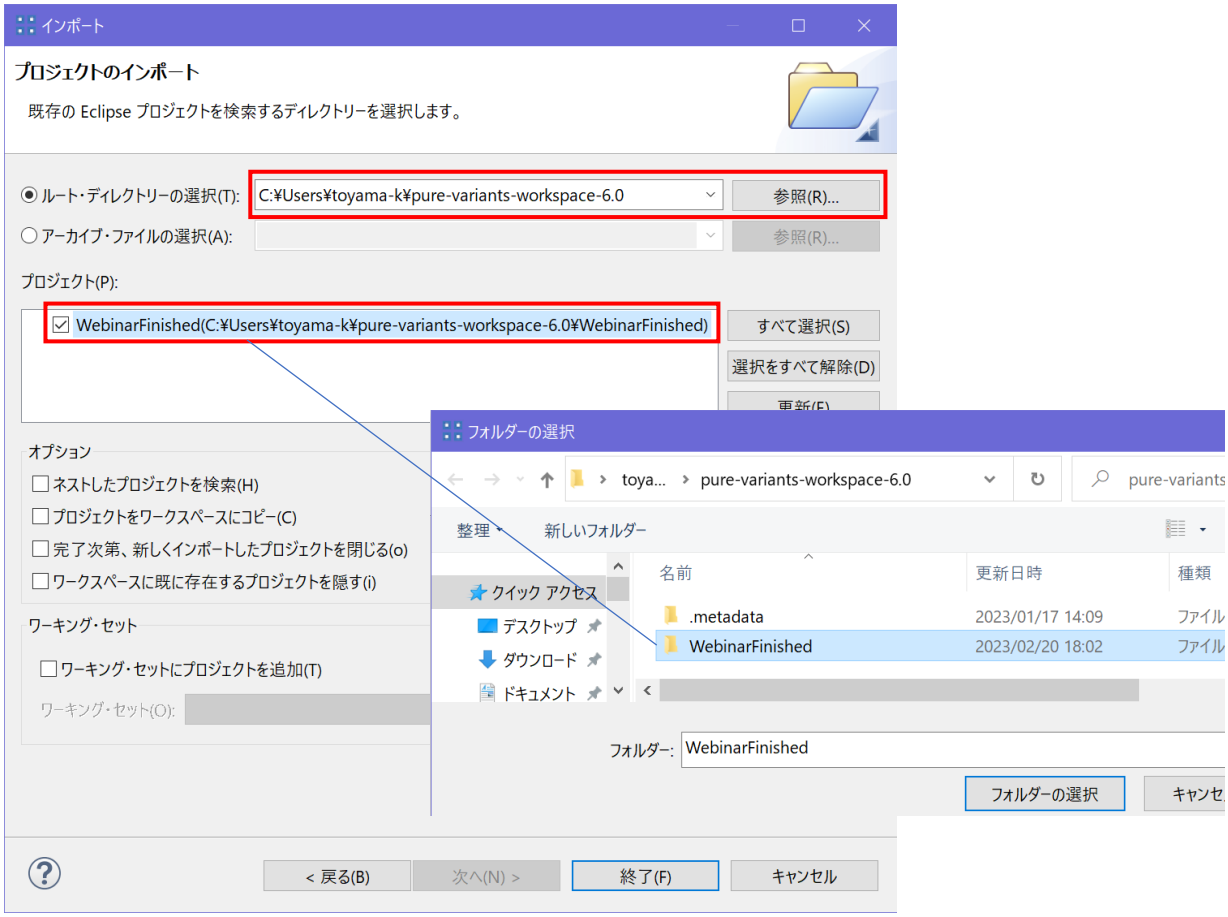
① メニューの ファイル > インポート でインポートウィザードを表示させます。また、Variant Projects ペイン内を右クリックしたコンテキストメニューでも表示可能です。



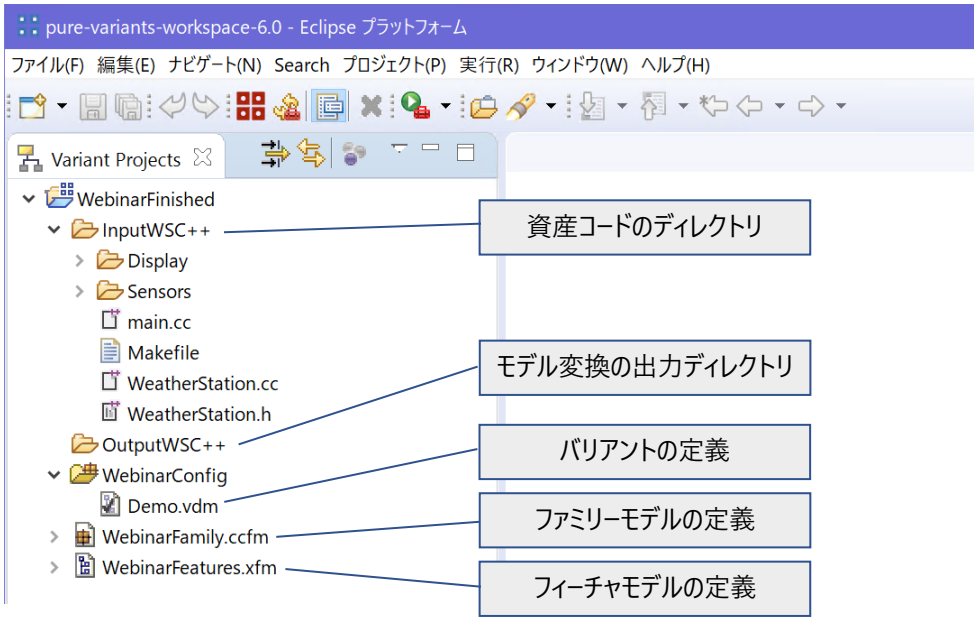
② インポートウィザードで 一般 > 既存プロジェクトをワークスペースへ を選択し、次へ をクリックします。



③ ルート・ディレクトリーの選択 欄に <ワークスペース> を設定します。参照 をクリックして フォルダの選択ウィザードで指定できます。

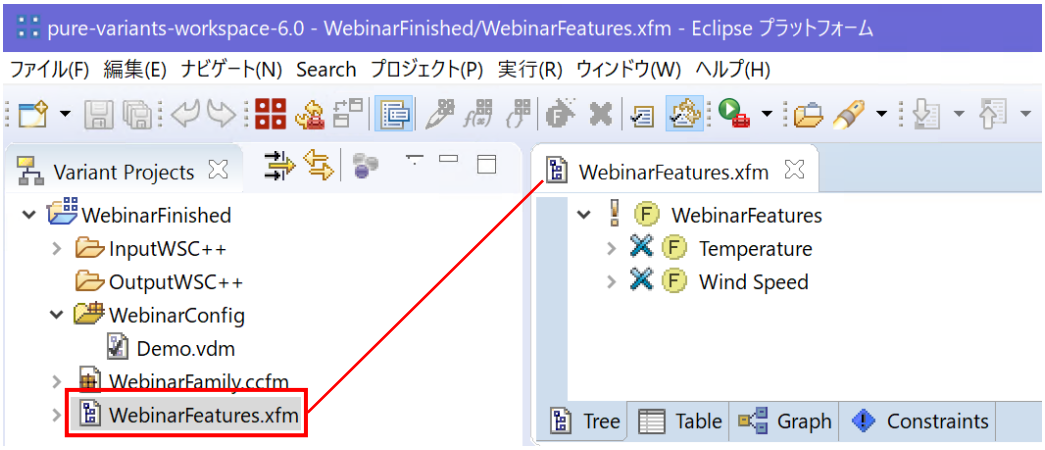


④ 準備でコピーした WebinarFinished がプロジェクト 欄に表示され、チェックボックスがチェックされていることを確認し、終了 をクリックします。以下のように pure::variants プロジェクト Webinar Finished がインポートされます。

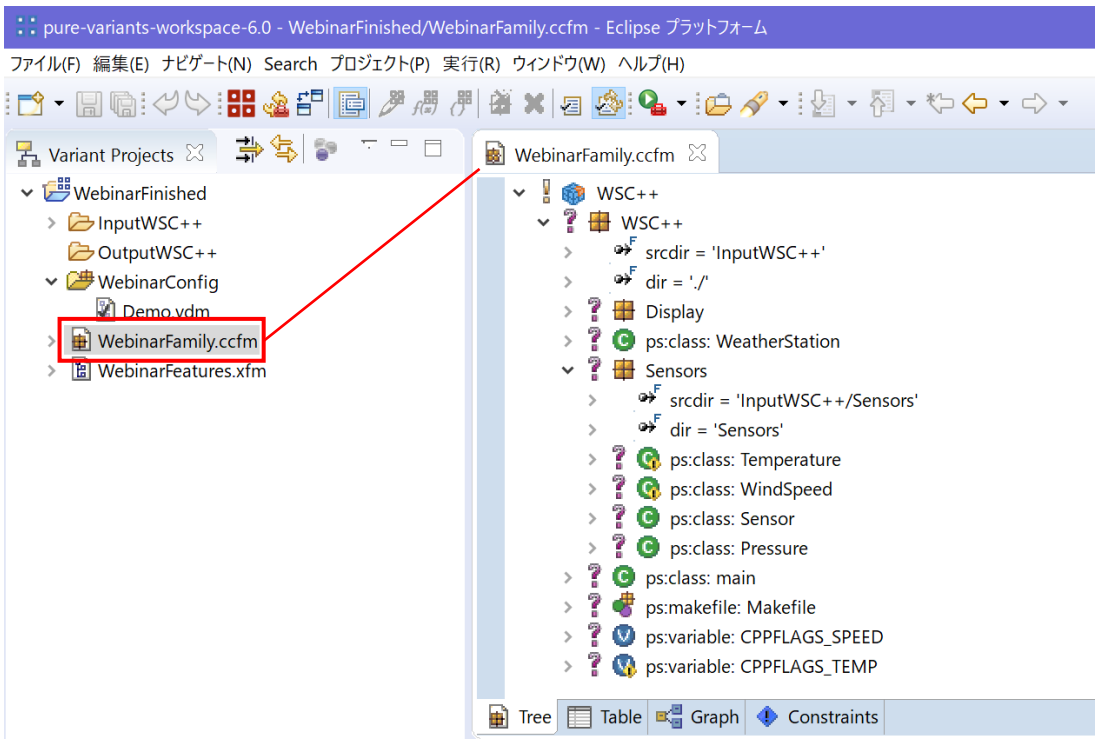


4. モデルの表示と製品バリアントの生成

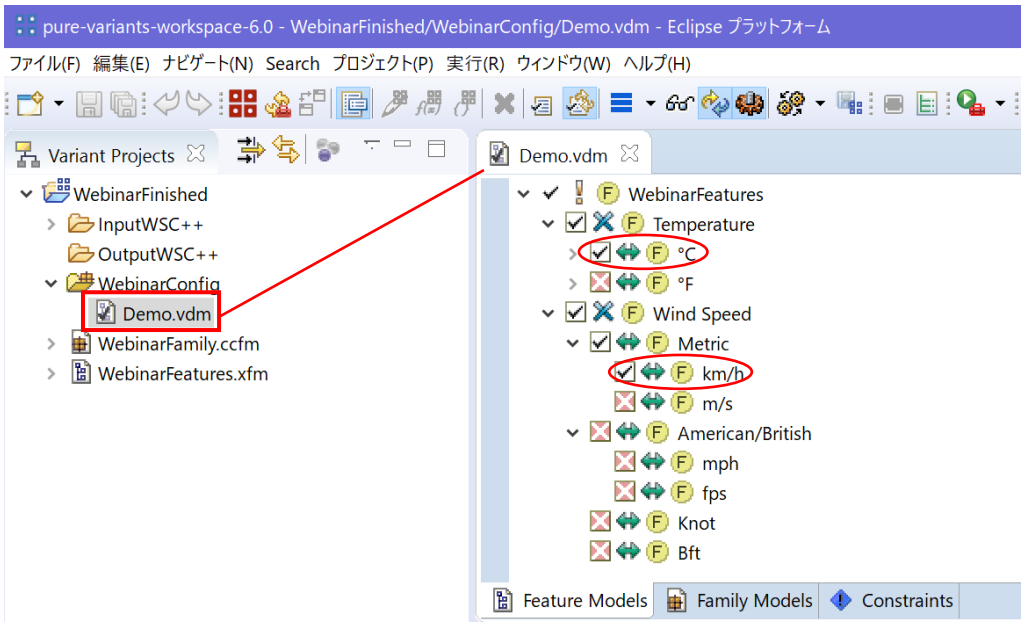
① WebinarFinished プロジェクトの WebinarFeatures.xfm をダブルクリックするとフィーチャモデルが表示されます。このモデルには、製品ファミリーの各機能やそれらの関係が階層構造で登録されています。各モデルエレメントの左端の「>」マークをクリックし、下位の階層を展開して表示できます。このモデルでは、気温を測定する Temperature 機能と、風速を測定する Wind Speed 機能があり、それぞれが複数の表示単位を持つことが分かります。



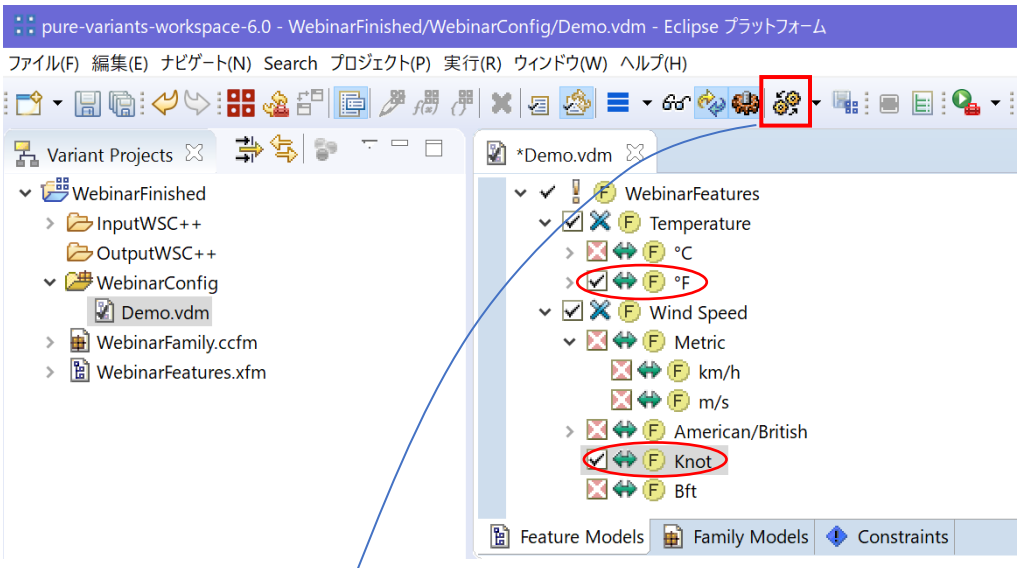
② 同様に WebinarFamily.ccfm をダブルクリックするとファミリーモデルが表示されます。このモデルは、製品ファミリーのクラス・オブジェクト・関数・変数・ドキュメントなどのコンポーネントで構成され、これらの配置や関係、制限、条件なども含みます。フィーチャモデルと同じく階層構造になっており、下位の階層を表示できます。ここでは srcdir などで、実際のソースコードや Makefile などへのリンクが登録されていることが分かります。これら 2 つのモデル間にはフィーチャの属性を参照するなどでリンクが張られており、製品ファミリーの変動要素が一貫して管理されます。



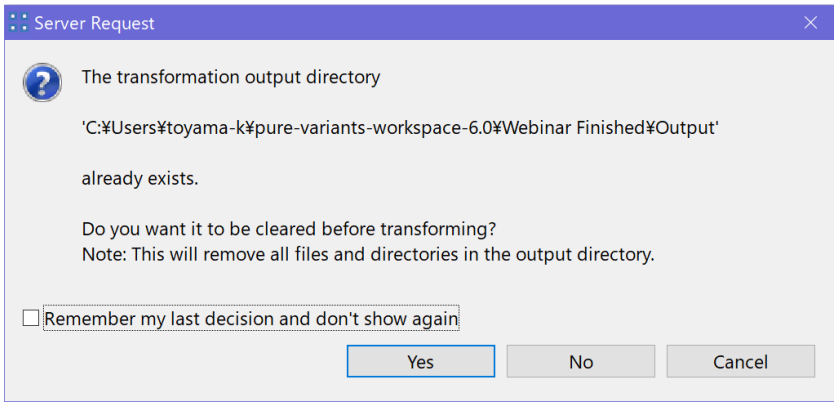
③ WebinarFinished プロジェクトのコンフィグレーションスペース WebinarConfig に、バリアント記述モデル（VDM）が登録されています。その中の Demo.vdm をダブルクリックしてモデルを開きます。このモデルで単一製品のフィーチャの組み合わせを選択できます。今ここでは、摂氏（ºC ）での温度表示と km/h での風速表示が選択されています。



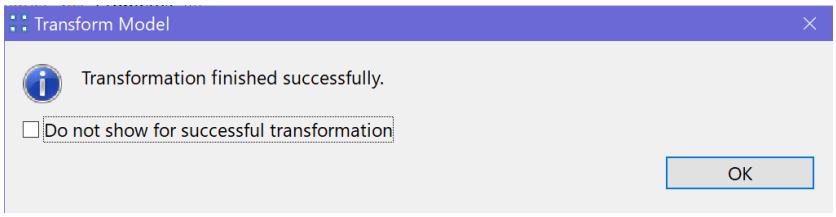
④ ここで、気温（Tempreture）の子フィーチャで摂氏（ºC）が選択されている（✓マーク）ものに対して、華氏（ºF）をチェックし、風速（Wind Speed）の子フィーチャで Knot をチェックすることで、華氏の温度表示とノットの風速表示を選択したバリアントが構成できます。
（ºF をチェックすると ºC の✓が ☒ に、Knot を✓すると km/h と Metric が ☒ に、それぞれ自動的に変更されます）



⑤ 上記の設定でモデル変換（Transform Model）ボタンをクリックすると、モデル変換によって、選択されたフィーチャに対応したファミリーモデルのコンポーネントが抽出されます。この例では、<ワークスペース>¥WebinarFinished¥Output¥WSC++ ディレクトリ以下に、ソースファイル、ヘッダファイルと Makefile が生成されます。（この生成場所は、モデル変換であらかじめ指定されるものです）

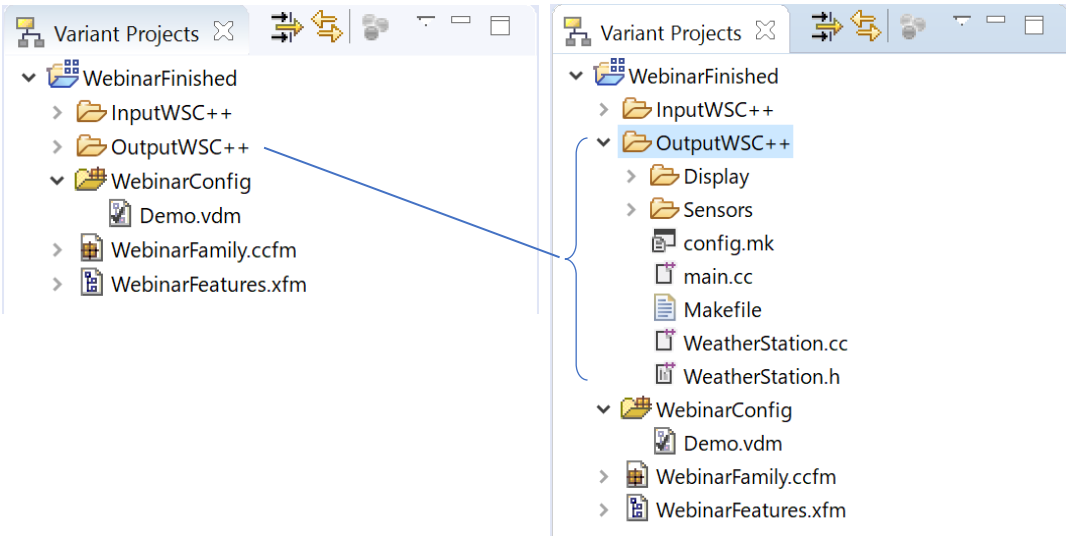


出力先にファイルが存在する場合は、上書きの確認ダイアログが表示されます。通常、Yes で進みます。



ファイル生成の終了ダイアログです。OK で進みます。

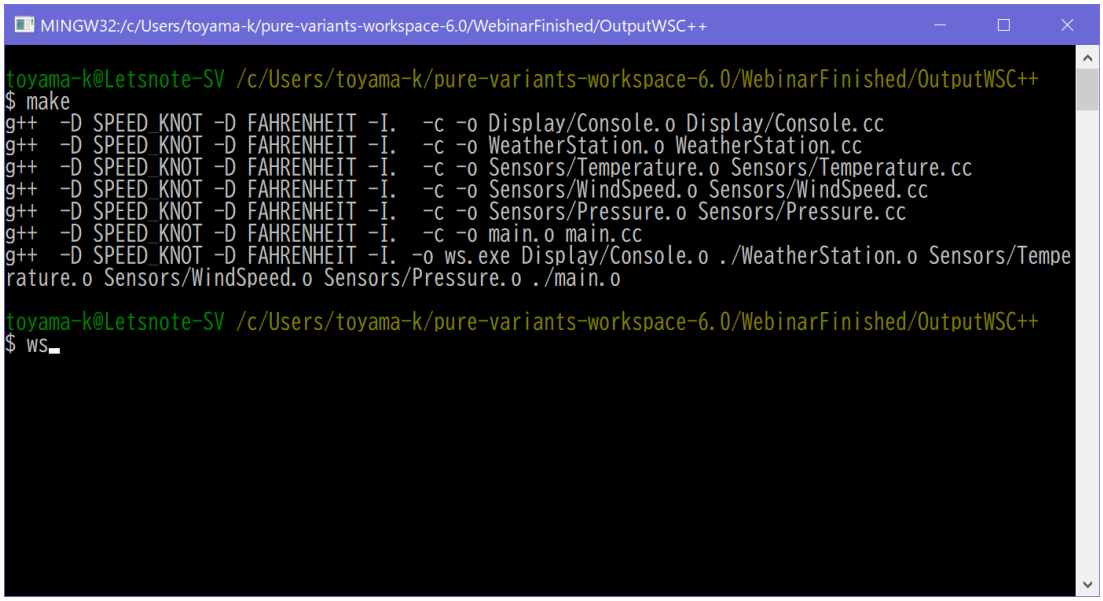
⑥ `OutputWSC++` 内に `Display` フォルダなどのファイルが生成されたことがわかり、その内容は下図右のようになります。 `InputWSC++` 内のファイルを使用して、このバリアントに対応したソースと Makefile が生成されています。
`make` コマンドは、この `OutputWSC++` ディレクトリにあるソース群を `g++` でコンパイルするもので、`config.mk` が Transform Model で生成されてバリアントごとに Makefile のコンパイルコマンドに対する定義を設定したものになります。

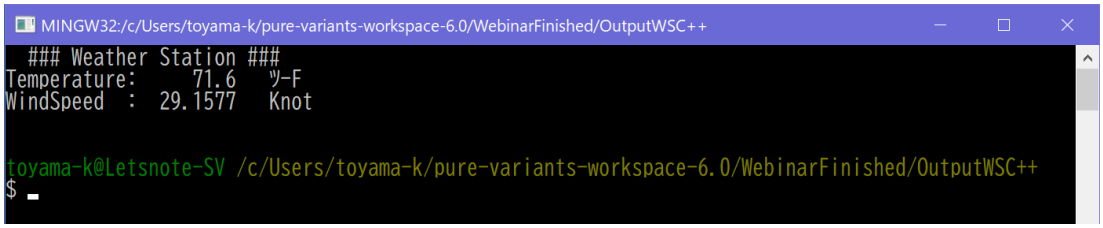


ファミリーモデル（`.ccfm`）とバリアントのソースファイルや Makefile の生成の関係については 6 章、7 章で説明します。

生成されたソースファイル群を実際にビルドします。MSYS の場合、MinGW のインストールディレクトリ以下にある[3] `msys.bat` を実行し、MinGW の実行コンソールを開きます。

⑦ コンソールで上記のソースディレクトリに移動し、`make` コマンドを実行することで、製品がビルドされ、実行ファイル（`ws.exe`）ができます。`ws.exe` を実行すると、気温と風力が計測されるアプリの動作が確認できます。（プログラムはデータを 20 回上書き表示して終了します）

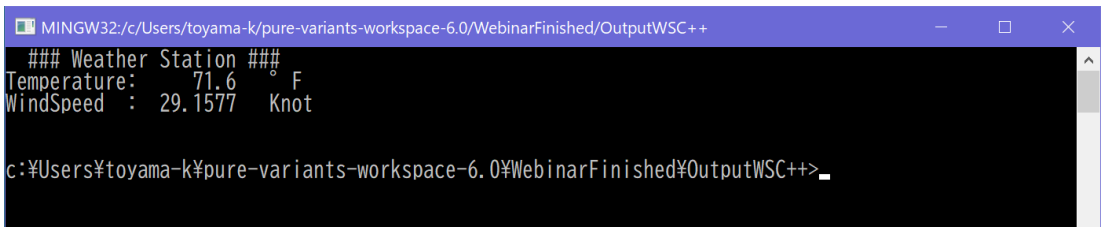




フィーチャで指定したとおり、華氏表示の気温とノット表示の風速が表示されています。

利用するコンソールによっては、文字コードエンコーディングとフォントの違いで出力の見え方が変わることがあります。（このソースは UTF-8 エンコーディングで記述されており、「ºF」の表示が異なります）

エンコーディングを UTF-8 にすると、華氏記号が表示できます[4]。

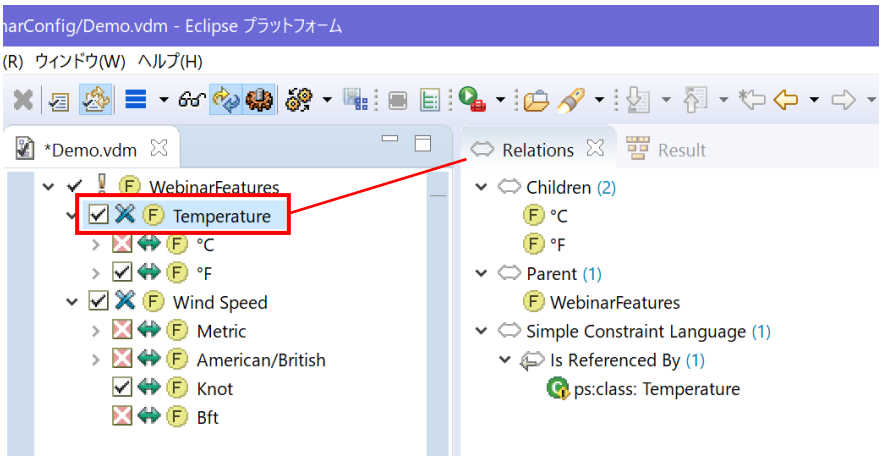


[3] `c:¥MinGW¥msys¥1.0¥` などとなります。

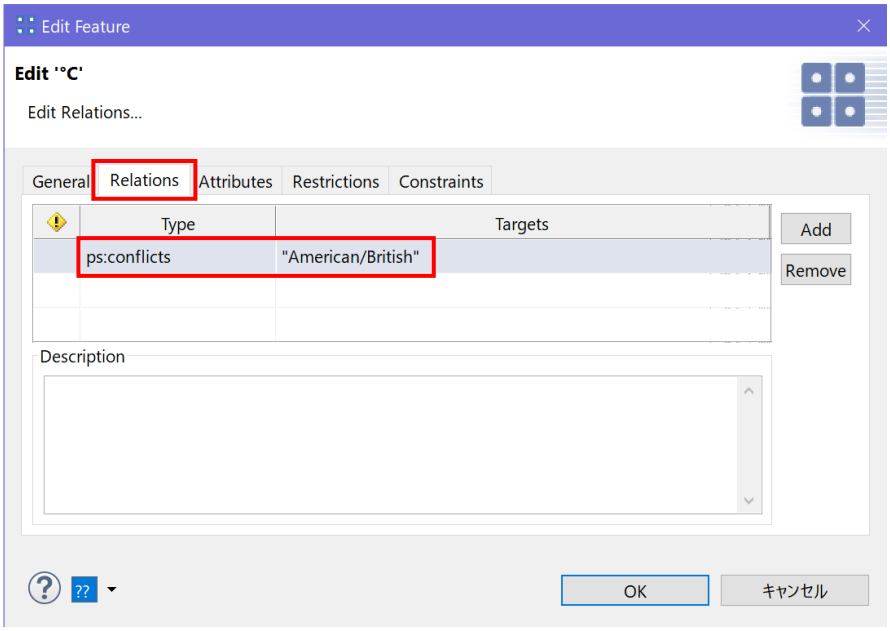
[4] MSYS のコンソールで UTF-8 とするには、`cmd.exe /K "chcp 65001"` を実行します。MSYS2 や WSL のコンソールは通常は UTF-8 です。（MSYS2 や WSL の場合、画面の初期化を行うため `make CONSOLE=UNIX` として make してください）

5. フィーチャとファミリーの関連付け

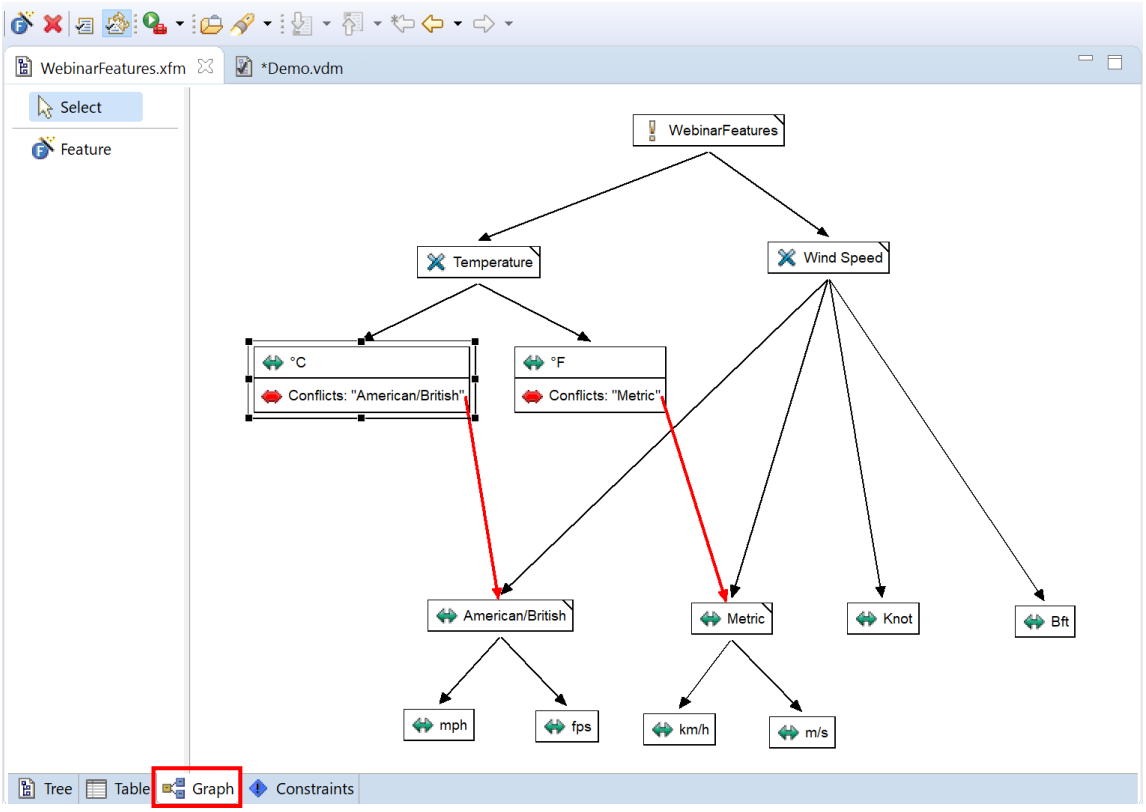
① フィーチャモデルの Temperature を選択すると、画面右端の Relations 表示に、関連付けられたコンポーネントが表示されます。ここでは、Temperature の子フィーチャとして °C と °F が、親フィーチャとして WebinarFeatures が設定されていることが分かります。また、ファミリーモデルで *ps:class* 属性の Temperature ファミリーエレメントがこのフィーチャを参照していることも分かります。
この Temperature ファミリーエレメントは温度計測の C++ 実装に対するものです。



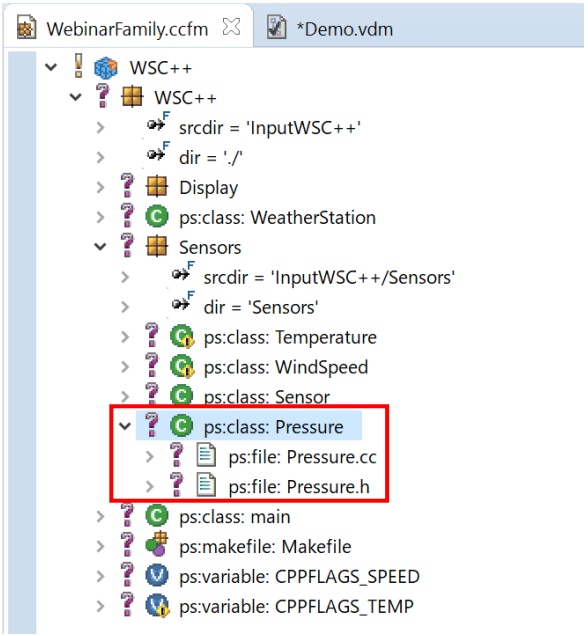
② このフィーチャ（ `.xfm` の °C ）のコンテキストメニューから Properties を選択することで、詳細な設定を確認できます。°C のプロパティを表示させて Relations タブを選択すると、American/British フィーチャに対して、*ps:conflicts* 型のリレーションが設定されています。これは、°C と American/British、2 つのフィーチャが同時には選択できない制約を表すリレーションの設定です。



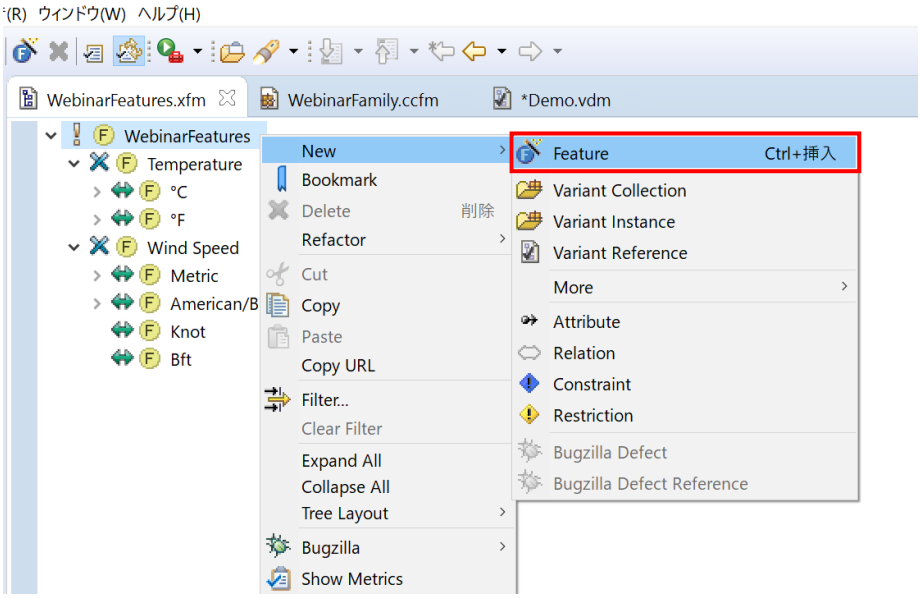
③ フィーチャモデルを表示するペインの下方の Graph タブでグラフィカルに表示することも可能です。



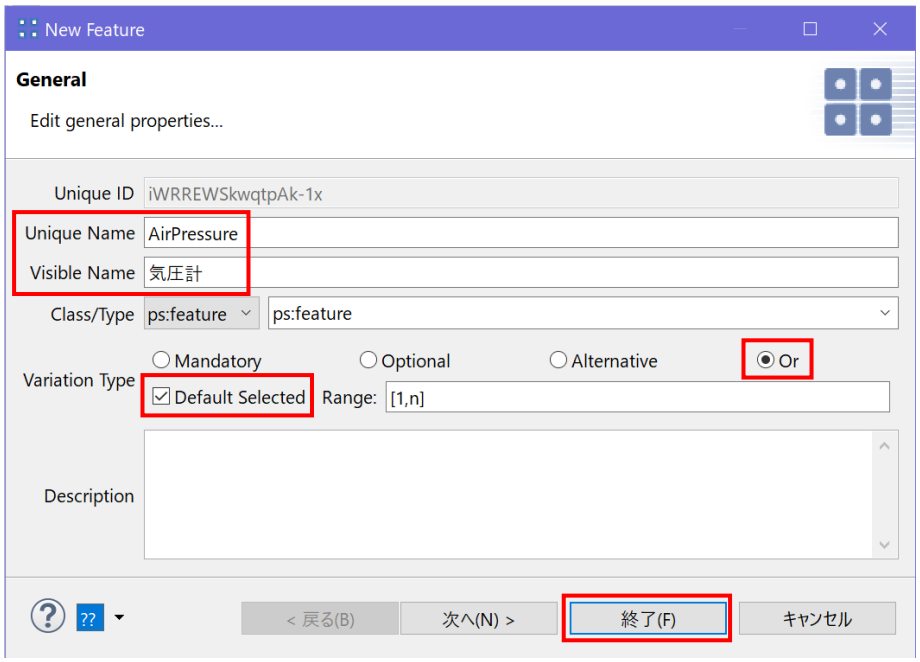
④ ファミリーモデルには資産コードとして、気圧計の実装クラス（*ps:class*: Pressure）が存在していますが、フィーチャモデル側には、気圧計はまだ登録されていません。そこで、これをフィーチャモデルに登録します。



⑤ WebinarFeature のコンテキストメニューで New > Feature を選択し、ウィザードを表示させます。

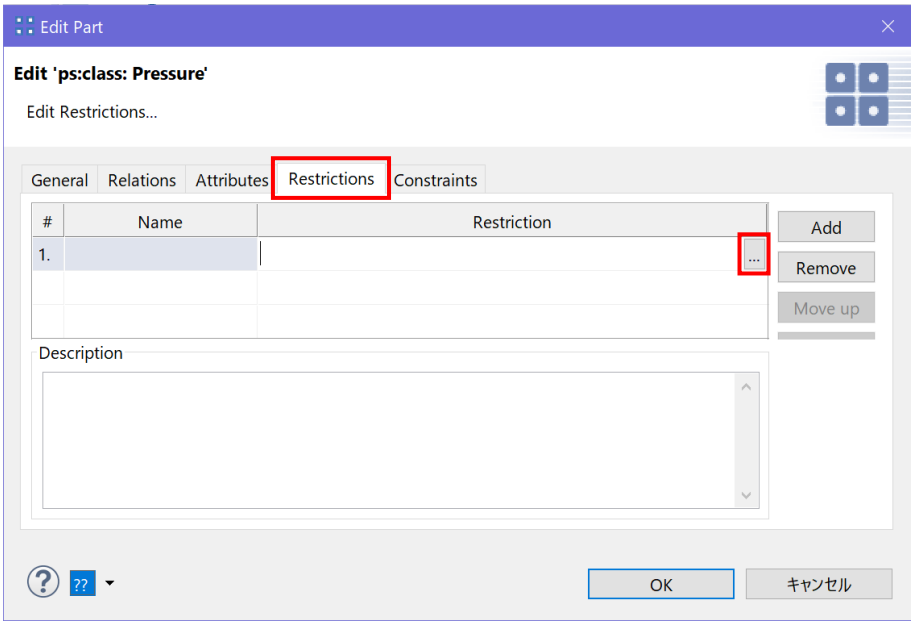


⑥ Unique Name として AirPressure、Visible Name として 気圧計 と設定します。Variation Type には、Mandatory（必須）、Optional（選択自由）、Alternative（どれか一つ）、Or（少なくとも一つ）がありますが、ここでは温度や風速と同様に、少なくとも一つ選択の Or を設定します。また、Default Selected のチェックも設定しておきます。

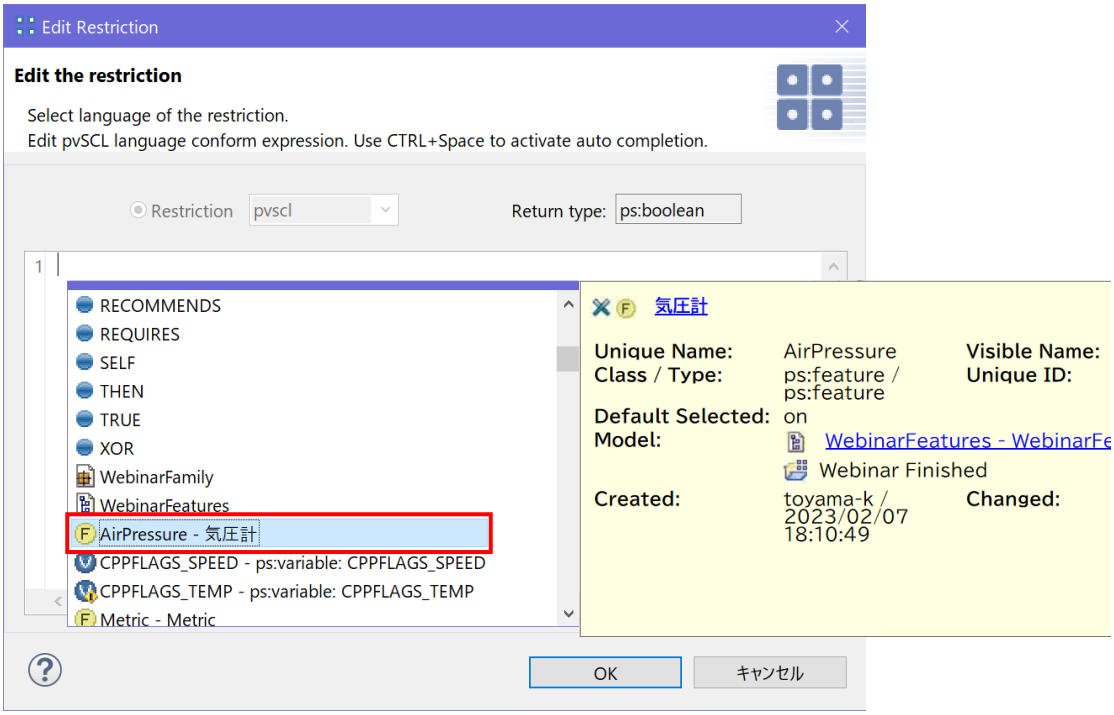


終了 でウィザードを閉じます。

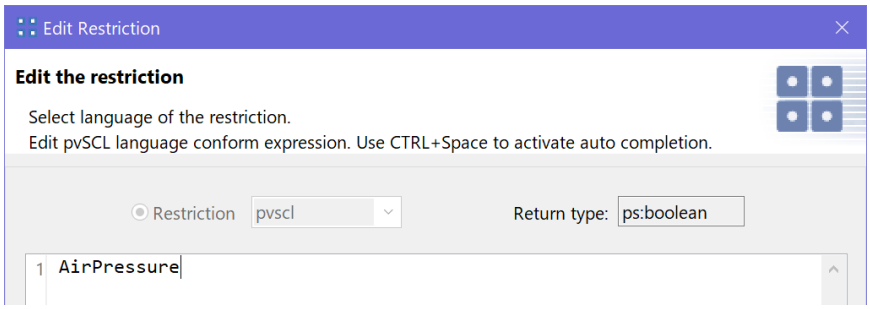
⑦ 追加した気圧計（AirPressure）フィーチャに、ファミリーモデルの対応するコンポーネントへのリンクを張ります。ファミリーモデル（.ccfm の Sensors）の *ps:class*:Pressure のコンテキストメニューで Properties を選択します。プロパティ編集ダイアログで Restrictions タブを選び、Restriction 欄の 1 行目をダブルクリックし、右端のアイコンをクリックします。これにより、気圧計を選択した際に Pressure の実装コードが関連付けられます。



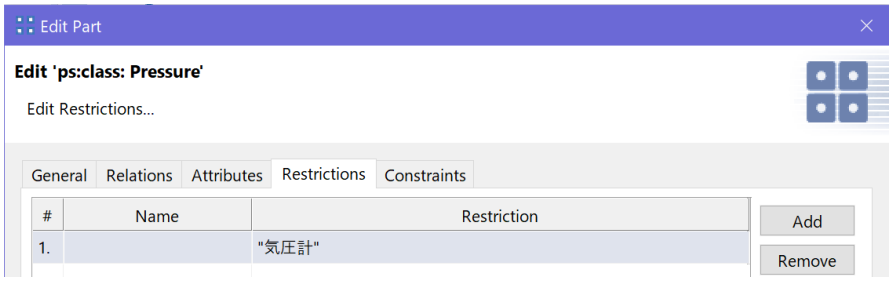
⑧ リストリクション編集ダイアログで、テキスト領域に文字が設定されていればそれを消去し、CTRL+Space を実行します。設定可能な項目のリストが表示されますので、先程フィーチャモデルに登録した「AirPressure - 気圧計」をダブルクリックします。



下図のようにテキスト領域に AirPressure が設定されます。

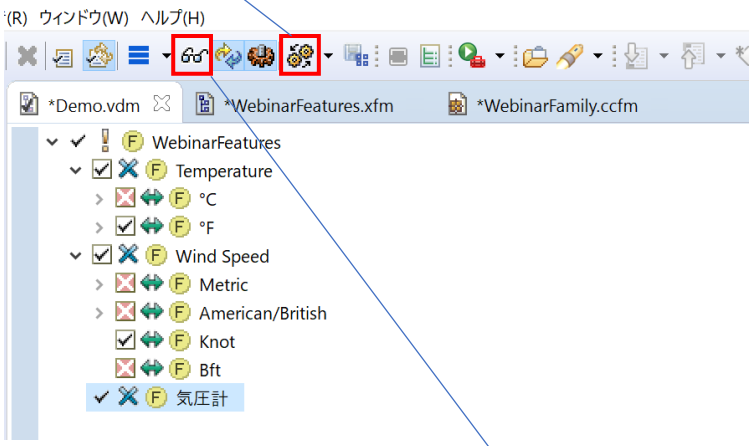


OK でこのダイアログを閉じます。



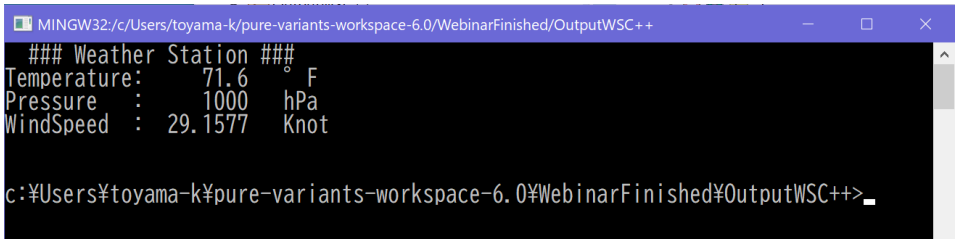
戻った⑦の Pressure 編集ダイアログで Restriction 欄に “気圧計” が設定されています。OK でダイアログを閉じます。

⑨ VDM を表示すると新たに「気圧計」が選択できるようになっています。ここで下図のように各フィーチャを選択し Transform Model を実行することで、気圧計を追加した新たな製品のソースコードが生成されます。



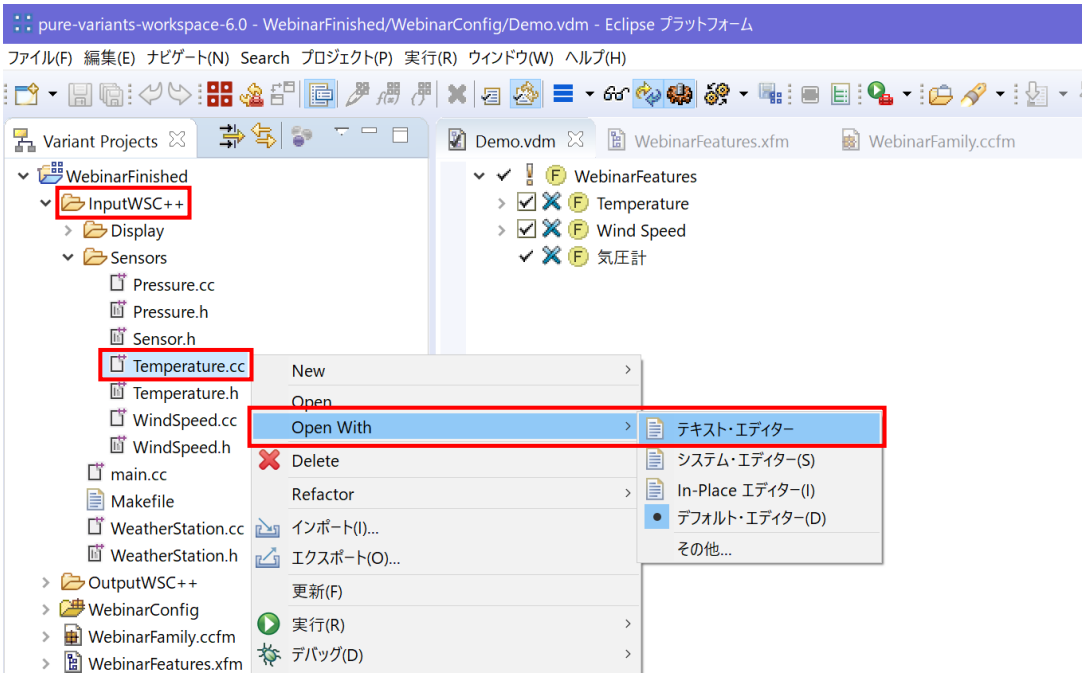
なお、「気圧計」選択時にエラーが発生する場合は、Evaluate Model を実行後、再度設定してください。

⑩ MSYS コンソールで再度ビルドを行い、プログラムを実行することで、気圧の計測が追加されていることが確認できます。

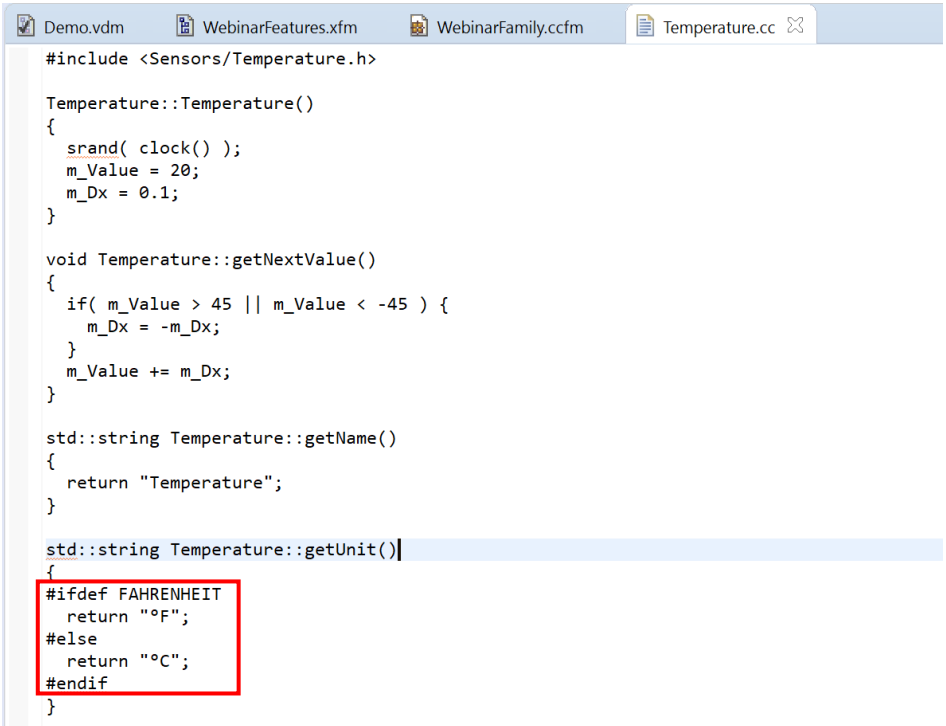


6. 条件付きコンパイル用のフラグの管理

① ファミリーモデルで管理される資産コードは、本例題の場合はプロジェクト内の InputWSC++ ディレクトリ以下にあります。たとえば、温度計のソースは Sensors¥Temperature.cc です。ファイル名のコンテキストメニューから Open with > テキスト・エディター でオープンできます[5]。

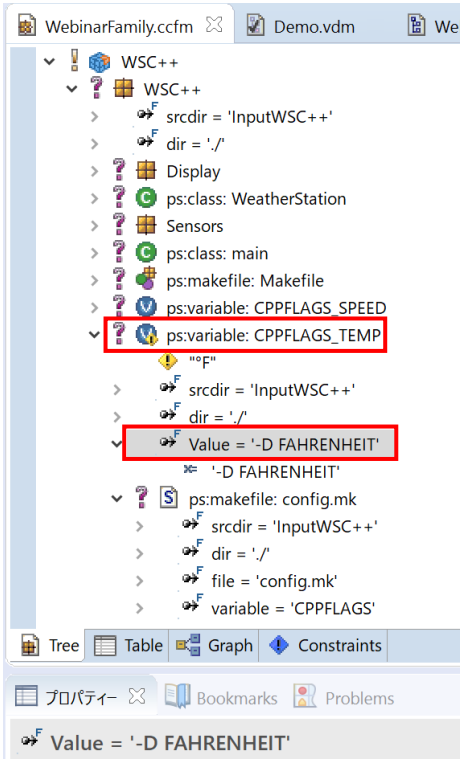


② このコードでは、#ifdef による条件付きコンパイルで FAHRENHEIT マクロ変数をフラグとして、摂氏/華氏の表示を切り替えています（Makefile 内で g++ コマンドのコンパイルフラグ変数 CPPFLAGS にこのマクロ変数定義を追加します）。これはフラグ単位でバリアビリティーを管理するものです。

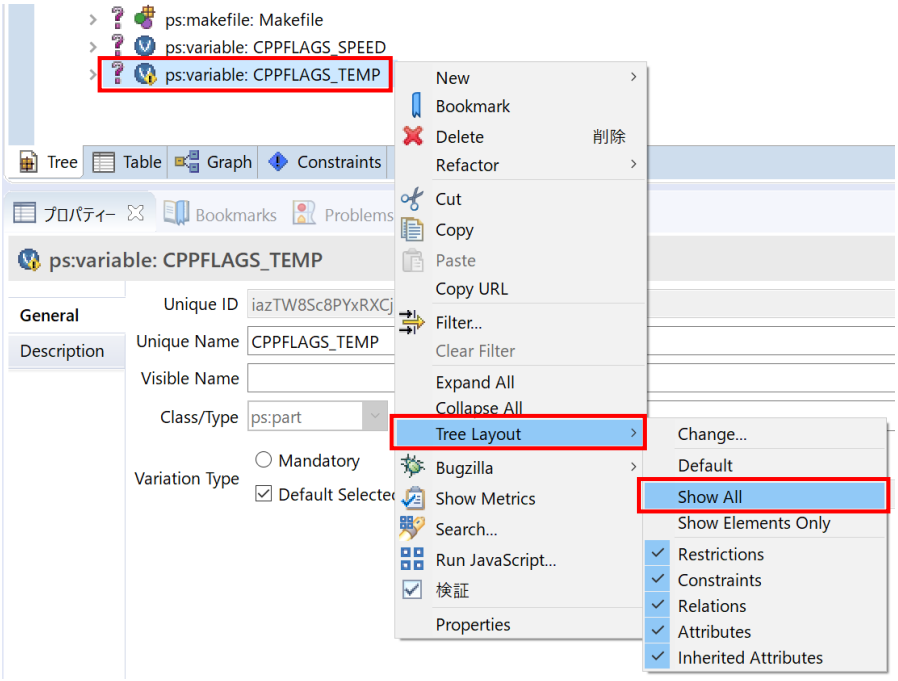


[5] システム・エディターを選択して外部のエディタを使用することもできます。

③ ファミリーモデルでは、WSC++ に ps:variable: CPPFLAGS_TEMP を定義し、CPPFLAGS に FAHRENHEIT マクロ変数を設定するコンパイルオプション '-D FAHRENHEIT' を Value に設定します。

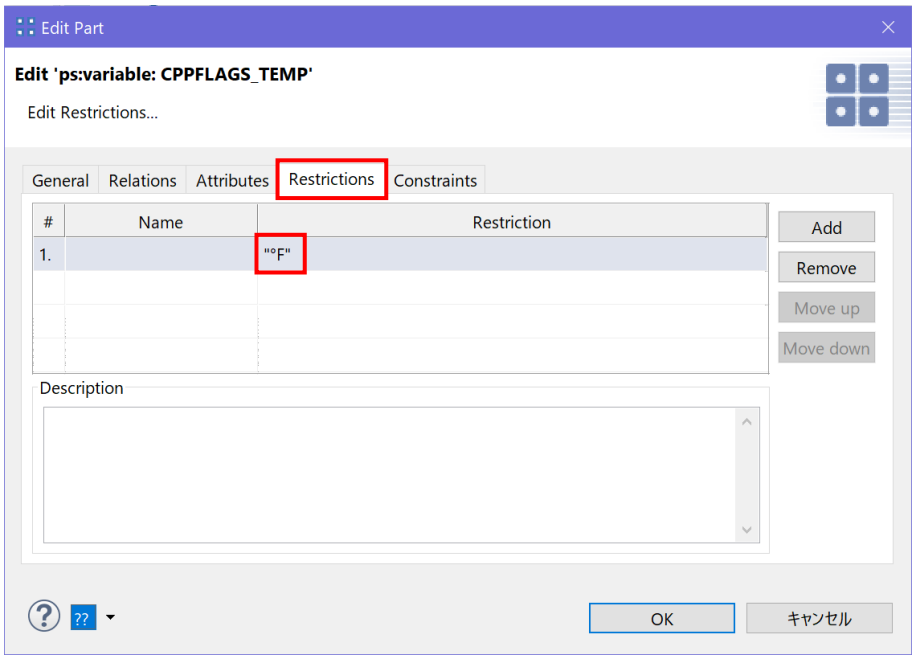


ここで、リストリクションなどが表示されない場合、下図のように ps:variable: CPPFLAGS_TEMP のコンテキストメニューから Tree Layout > Show All を選択すると、設定の詳細が表示されるようになります。



④ このコンパイルオプション `-D FAHRENHEIT` は、`Makefile` 内で `config.mk` をインクルードして指定されます。
リストリクションで °F と関係づけられているので、°F フィーチャが指定されている場合に、モデル変換によってこの CPPFLAGS_TEMP の値（ `-D FAHRENHEIT` ）が、コンパイルフラグ変数 CPPFLAGS に対するものとして `config.mk` ファイルに出力されます。

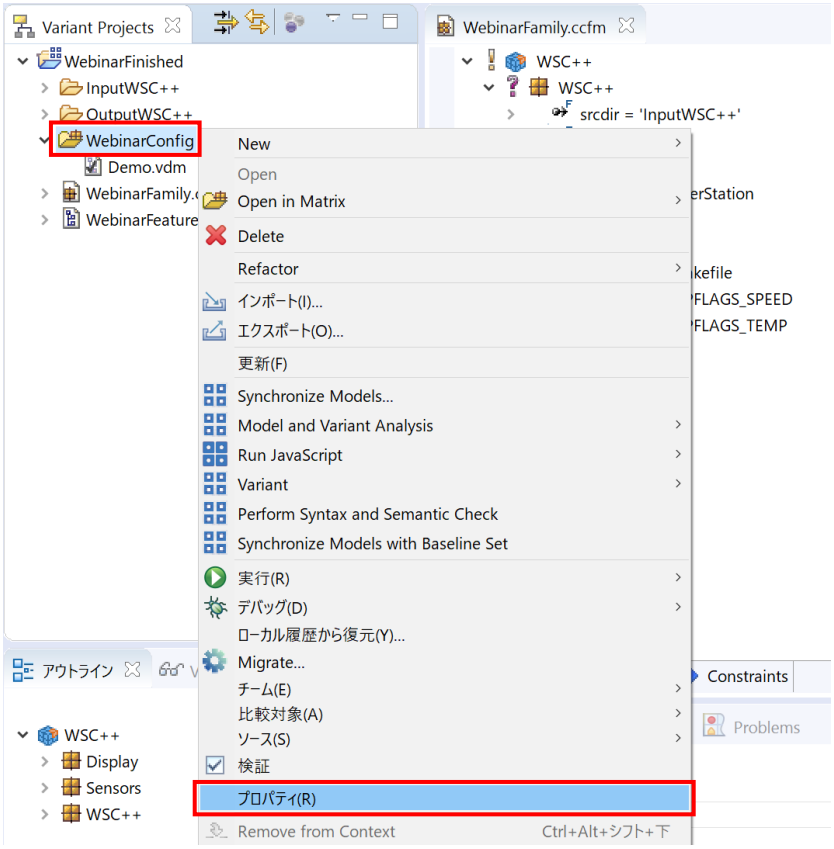
ps:variable: CPPFLAGS_TEMP をダブルクリックしてプロパティを表示し、Restrictions タブをクリックすると TemperatureF フィーチャ（°F）が選択された時のみ有効になる様にリストリクションが設定されていることが確認できます。



7. モデル変換処理による Makefile の生成

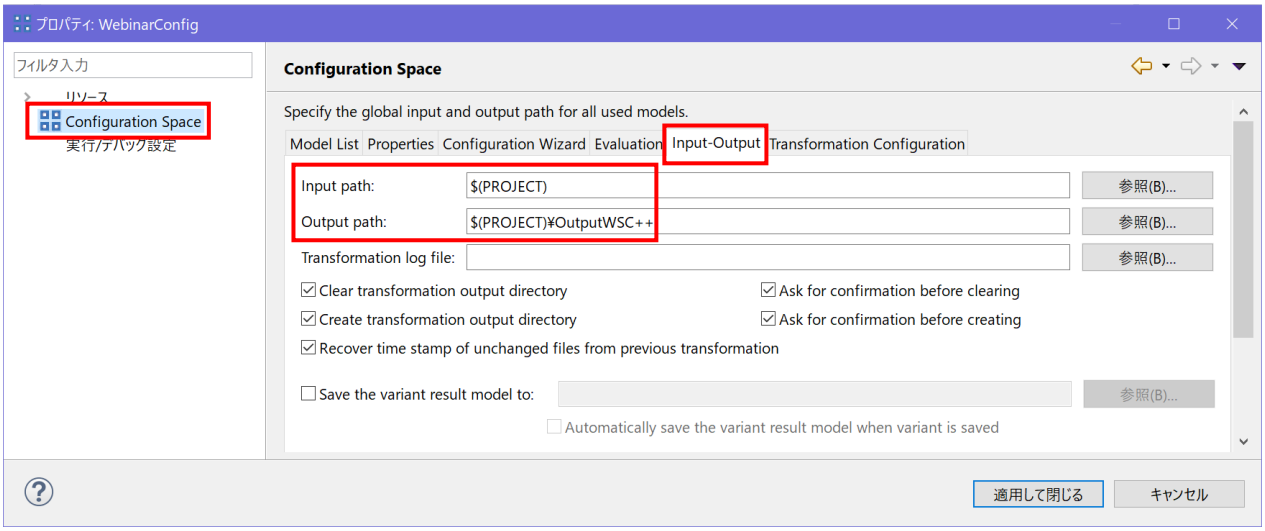
モデル変換（Transformation）によってバリアントに固有の Makefile を生成し、資産である InputWSC++ のソースファイル（.cc と.h）からバリアントの実行ファイルを生成する手順は以下です。

① Transformation の設定は、Configuration Space を使って行います。WebinarConfig のコンテキストメニューで プロパティ を選択します。

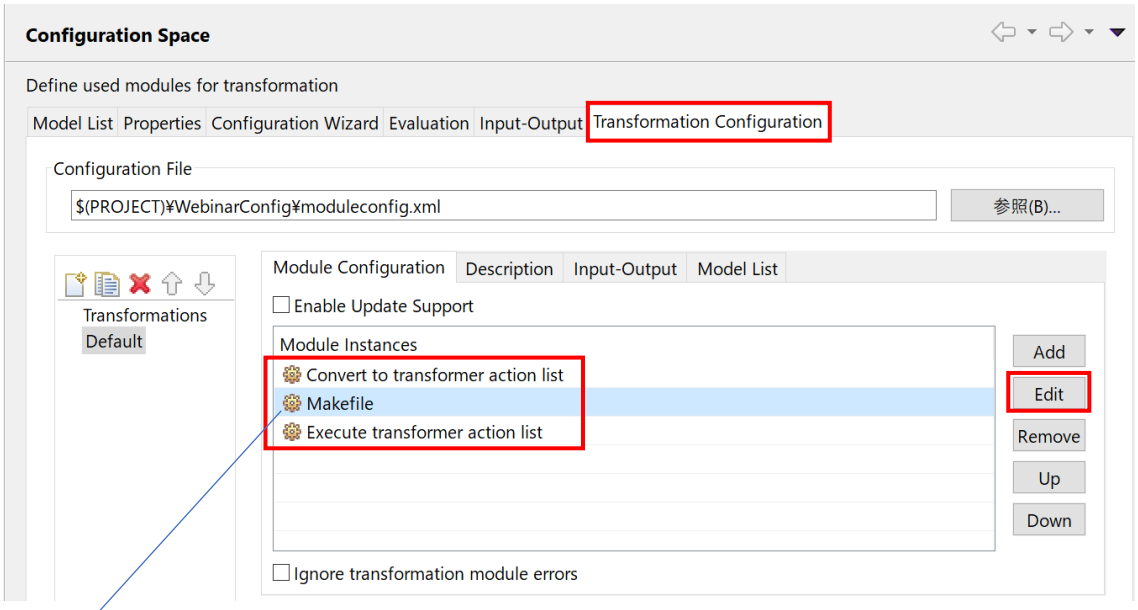


② プロパティ ダイアログで Configuration Space を選択し、Input-Output タブを選択します。ここでは、Input path としてソースファイルのあるディレクトリ（ここでは WebinarFinished プロジェクトのディレクトリ $(PROJECT)：`WebinarFinished`）、Output Path として VDM をモデル変換して生成されるファイルの出力先（ここでは $(PROJECT)¥OutputWSC++：`WebinarFinished¥OutputWSC++`）が指定されています。

Transform Model を実行すると、この設定に基づいて必要なファイルがコピーされ、必要に応じて変換されます。たとえば、p.5 ②の WebinarFamily.ccfm で WSC++の srcdir=InputWSC++と設定されている入力パスは `WebinarFinished¥InputWSC++` となり、dir=./ と設定されている出力パスは `WebinarFinished¥OutputWSC++` となります。

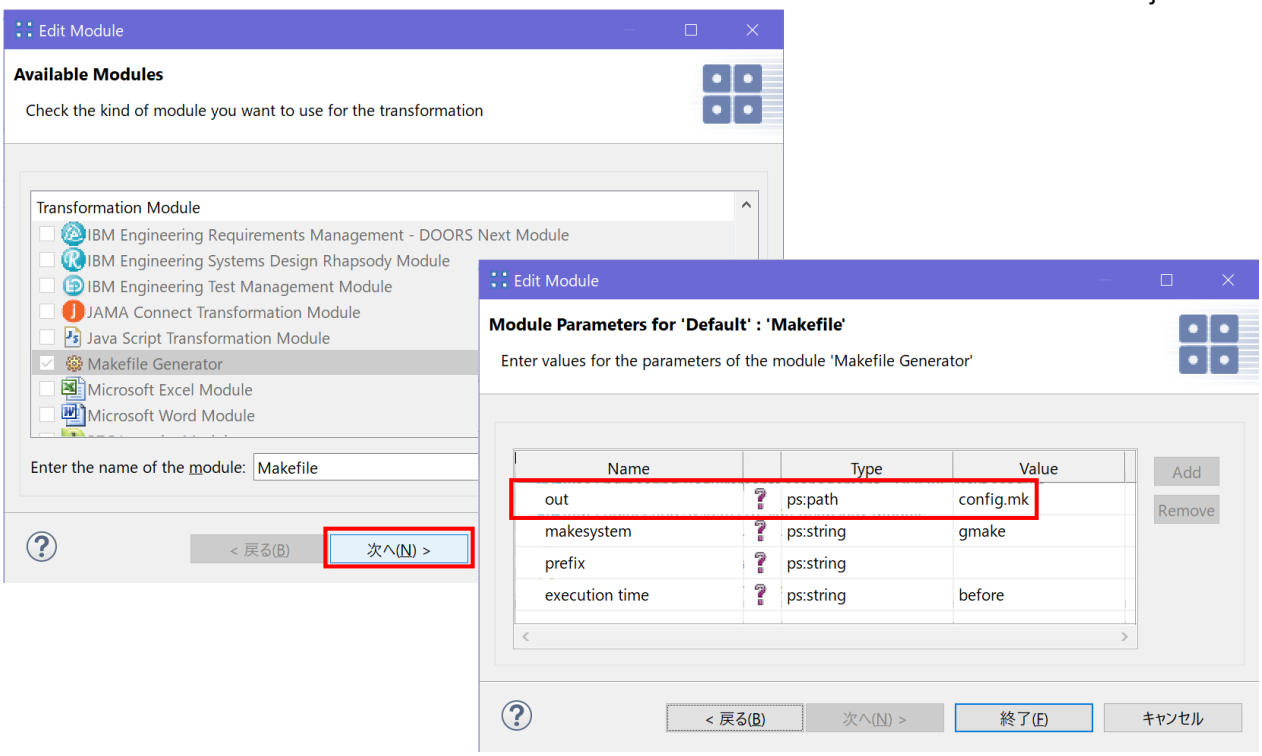


③ Transformation Configuration タブをクリックすると、Transform Model 実行時に行う動作の規定が確認できます。Convert to transformer action list でコピーするファイルのリストを作成し、Execute transformer action list でそれらのファイルを実際にコピーし、Makefile でファイルリストを基に Makefile を作成しています。



Makefile を選択して Edit をクリックし、モジュール編集ダイアログを表示します。

④ モジュール編集ダイアログで 次へ をクリックして Makefile Generator の詳細な設定が確認できます。出力するファイル名は、out のパラメータで config.mk と指定されており、先の ② の Input-Output タブで Output Path として指定したディレクトリに作成されます。ここでは設定することが目的ではないので、キャンセル し、戻ったダイアログで 終了 します。



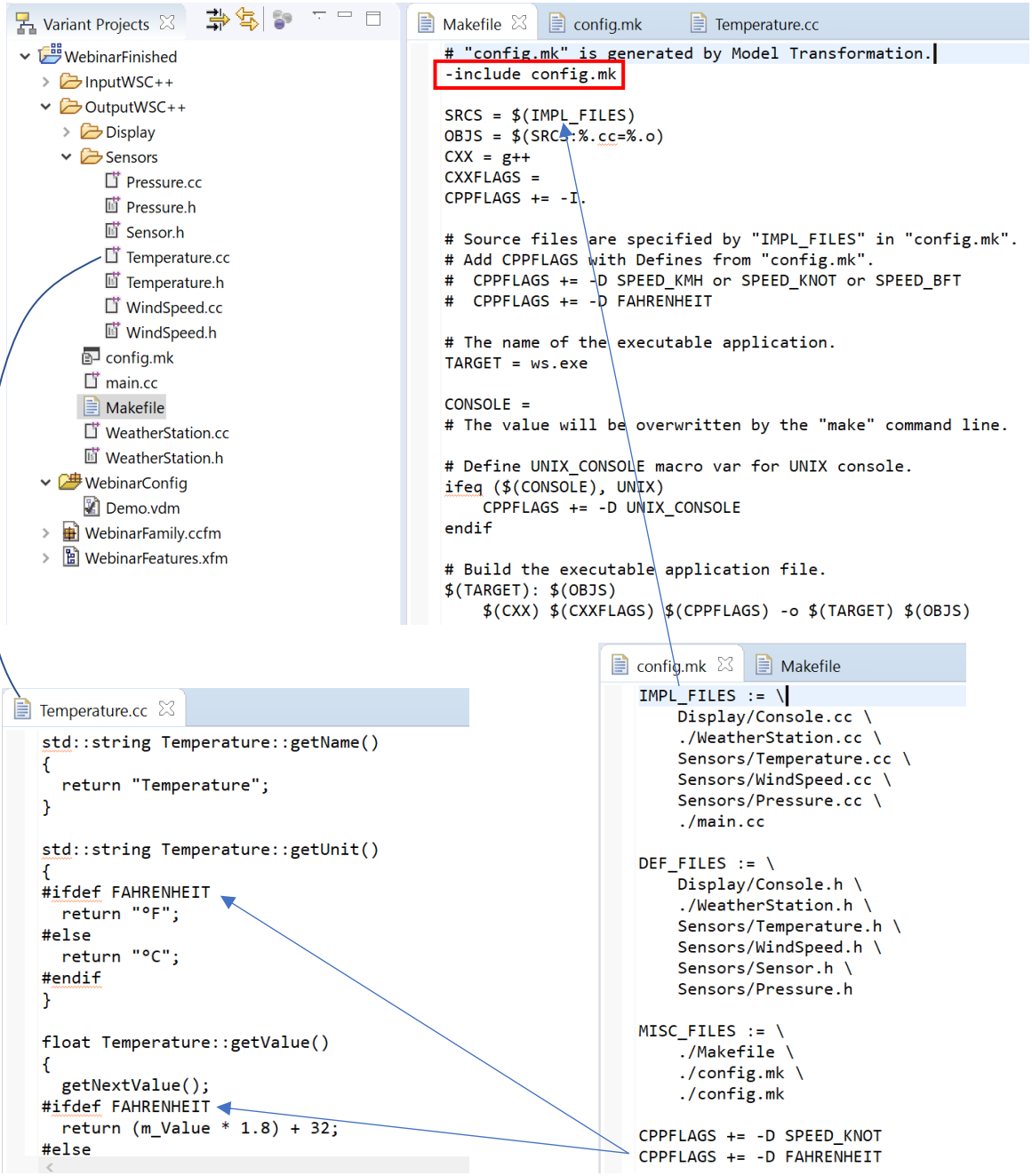
Makefile Generator によって `config.mk` ファイルには、ファミリーモデルでの定義から以下のように、入力となるソースファイル（`IMPL_FILES`）やヘッダファイル（`DEF_FILES`）を抽出した[6]ものが生成され、`Makefile` 内で `$(IMPL_FILES)` として参照するなどしてソースファイルを指定できます。さらに、プリプロセッサオプション変数 `CPPFLAGS` への設定が生成されます。

config.mk
```
IMPL_FILES := \
    Display/Console.cc \
    ./WeatherStation.cc \
    Sensors/Temperature.cc \
    Sensors/WindSpeed.cc \
    Sensors/Pressure.cc \
    ./main.cc

DEF_FILES := \
    Display/Console.h \
    ./WeatherStation.h \
    Sensors/Temperature.h \
    Sensors/WindSpeed.h \
    Sensors/Sensor.h \
    Sensors/Pressure.h

MISC_FILES := \
    ./Makefile \
    ./config.mk \
    ./config.mk

CPPFLAGS += -D SPEED_KNOT
CPPFLAGS += -D FAHRENHEIT
```

⑤ Variant Projects で WebinarFinished > OutputWSC++ > Makefile のコンテキストメニューから Open With > テキスト･エディタ で、生成された `Makefile` を確認できます。このファイル内で、以下の様に `config.mk` がインクルードされており、コンパイルオプションへのマクロ変数の定義が追加されるよ

6 ファミリーモデルで type が impl や def、misc であるものをそれぞれまとめます。

うになっています。



make での g++のコンパイルオプションに -D FAHRENHEIT が指定されるので、Tempreture.cc などのソースコードで条件付きコンパイルの「華氏」が有効になります。
また、同様に -D SPEED_KNOT で WindSpeed.cc での単位が「ノット」として設定されます。

富士設備工業株式会社 電子機器事業部
〒591-8025 大阪府堺市北区長曽根町1928-1
Tel: 072-252-2128 www.fuji-setsu.co.jp