重複する翻訳メモリとは

同じ原文に対して複数の訳文が登録されているメモリ。状況に応じて訳文を使い分けるために翻訳者が意図的に複数の訳文を登録している場合 (**意図的な重複、必要な重複**) と、複数の翻訳者で分業している場合などに、メモリスワップの結果、もう使用していない古い訳文が修正／削除されることなく残ってしまう場合 (**意図しない重複、不要な重複**) があります。

I. 意図的な重複（必要な重複）

CASE 1. クライアントから指定されたスタイルの仕様上、1 つの用語に何通りかの訳出が必要

例えば、ソフトウェアはローカライズせず、マニュアルのみを翻訳するプロジェクト

A). 対象製品の UI の訳出は [English (日本語訳)] というスタイルが指定されている

```
{0>Click Finish to close the dialog.<}0{>[Finish (完了) ] をクリックしてダイアログを閉じます。<0}
```

B). 製品のインストール手順などで、Windows の UI が登場。たまたま同じ単語だった

```
{0>Click Finish to close the dialog.<}0{>[完了] をクリックしてダイアログを閉じます。<0}
```

CASE 2. 原文が FrameMaker 形式で、索引マーカテキストの訳出が必要

日本語の場合、索引マーカテキストには<:so>タグを使って読み仮名を入力する必要があります。

A). 章見出し

```
<ps "heading 2" 38>{0>Generating a Pie Chart<}0{>円グラフの作成<0}
```

B). 章見出しと同じ原文が索引にも使われている

```
<imk 2>
<ie>{0>Generating a Pie Chart<}100{>円グラフの作成<:so>えんぐらふのさくせい<0}
</imk>
```

II. 意図しない重複（不要な重複）

CASE 1. 翻訳作業を開始してしまってから、当初指定されていた訳語に変更があった場合

自分の担当部分以外で修正が行われている場合、修正前のメモリと修正後のメモリをいずれも Merge／結合インポートしているため、修正前のメモリもそのまま残り、複数候補が発生します。

A). **当初は、UI の翻訳がまだ確定していないので、仮に英語のままにしておくように指示があった**

```
{0>Click Preference<}100{>[Preference] をクリックします。<0}
```

B). **プロジェクト開始後、クライアントより訳が確定した UI リストが提供され、訳語の変更を指示された**

```
{0>Click Preference<}100{>[環境設定] をクリックします。<0}
```

CASE 2. 細かいスタイルの指定が事前になかったために、同じ原文に訳者によって異なる訳出が発生

後で気が付いて表現を統一したとしても、すでに Merge／結合インポート済みのメモリについては、TM の使用者が明示的に削除しない限り、修正前の訳文もそのまま残り、複数候補が発生します。

A). **翻訳者** A

```
{0>See <:xr "ÒSetting Up ClustersÓ on page<:hs>354" 2> for more information on setting up clusters.<}0{>クラスタの設定方法の詳細については、<:xr "ÒSetting Up ClustersÓ on page<:hs>354" 2>を参照してください。<0}
```

B). **翻訳者** B

```
{0>See <:xr "ÒSetting Up ClustersÓ on page<:hs>354" 2> for more information on setting up clusters.<}100{>クラスタの設定については、<:xr "ÒSetting Up ClustersÓ on page<:hs>354" 2>を参照してください。<0}
```

CASE 3. 他の翻訳者の訳文に入力ミスがあったが、後になってから本人が気づいて修正している

修正した本人のメモリでは、同じ訳文に修正が加わりますが、修正前のメモリをすでにインポートしてしまってから、修正後のメモリを Merge／結合オプションでインポートしている他の翻訳者のメモリでは、修正前の訳文も削除されないまま残ります。

A.) **入力ミス**

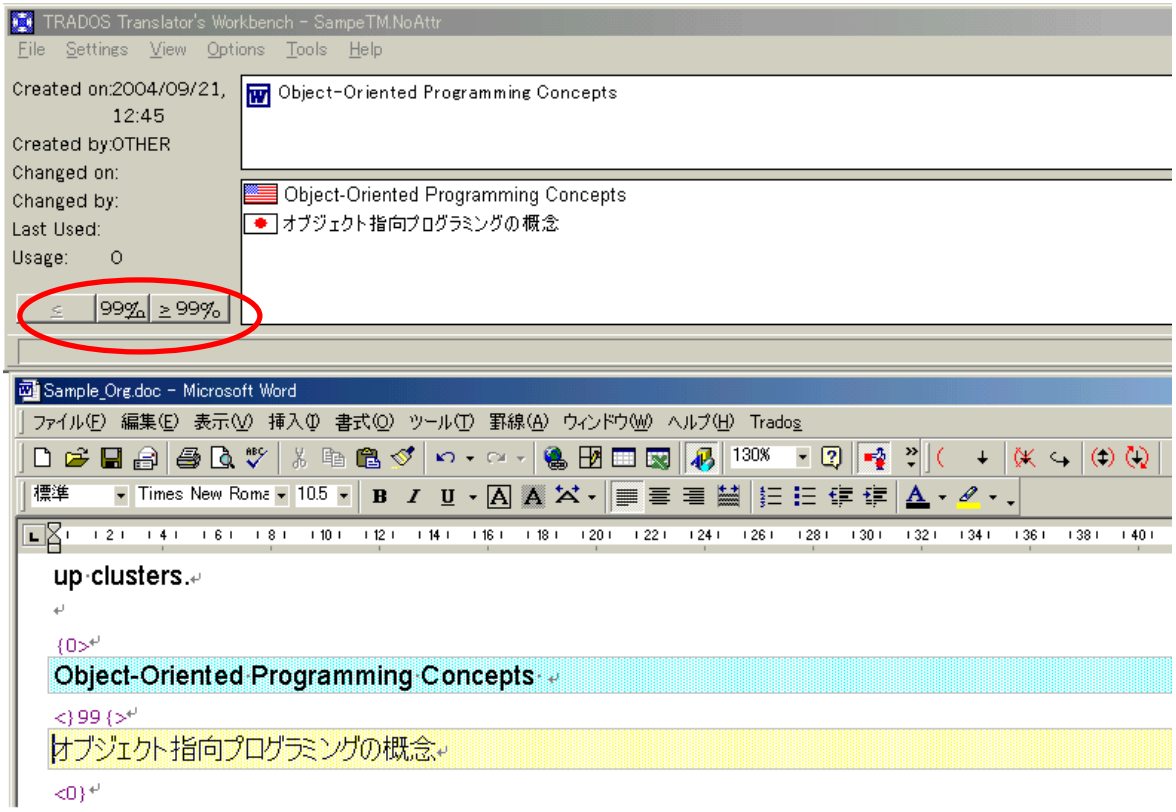
```
{0>Object-Oriented Programming Concepts <}0{>オブジェクト思考プログラミングの概念<0}
```

B.) **後に修正**

```
{0>Object-Oriented Programming Concepts <}0{>オブジェクト指向プログラミングの概念<0}
```

不要な重複が蓄積した翻訳メモリの例

CASE 1. 他の翻訳者の訳文に入力ミスがあったため、修正前後のメモリが混在している (II-CASE3)



実際にはこの原文に対応する訳文は 1 つしか存在しないはずなのに、入力ミスを修正する前のメモリがそのまま残っているために複数の訳文が存在しています。複数の訳文の存在はメモリオプションのデフォルトで 1%のペナルティが付くので、後でこのメモリを使って自動翻訳をしても、この訳文は完全一致 (100%) と見なされず、翻訳者が手動で取得する作業が必要になります。

CASE 2. 細かいスタイルの指定が事前になかったために、同じ原文に訳者によって異なる訳出が発生

A). **翻訳者** A

{0>Presently, the majority of administrative tasks must be performed using the administration user interface, part of the Web UI.<}0{>現時点では、管理タスクのほとんどは、Web UI に含まれる管理ユーザー インターフェイスから行う必要があります。<0}

B). **翻訳者** B

{0>Presently, the majority of administrative tasks must be performed using the administration user interface, part of the Web UI.<}0{>現時点では、管理タスクのほとんどは、Web UI に含まれる管理ユーザインタフェースから行う必要があります。<0}

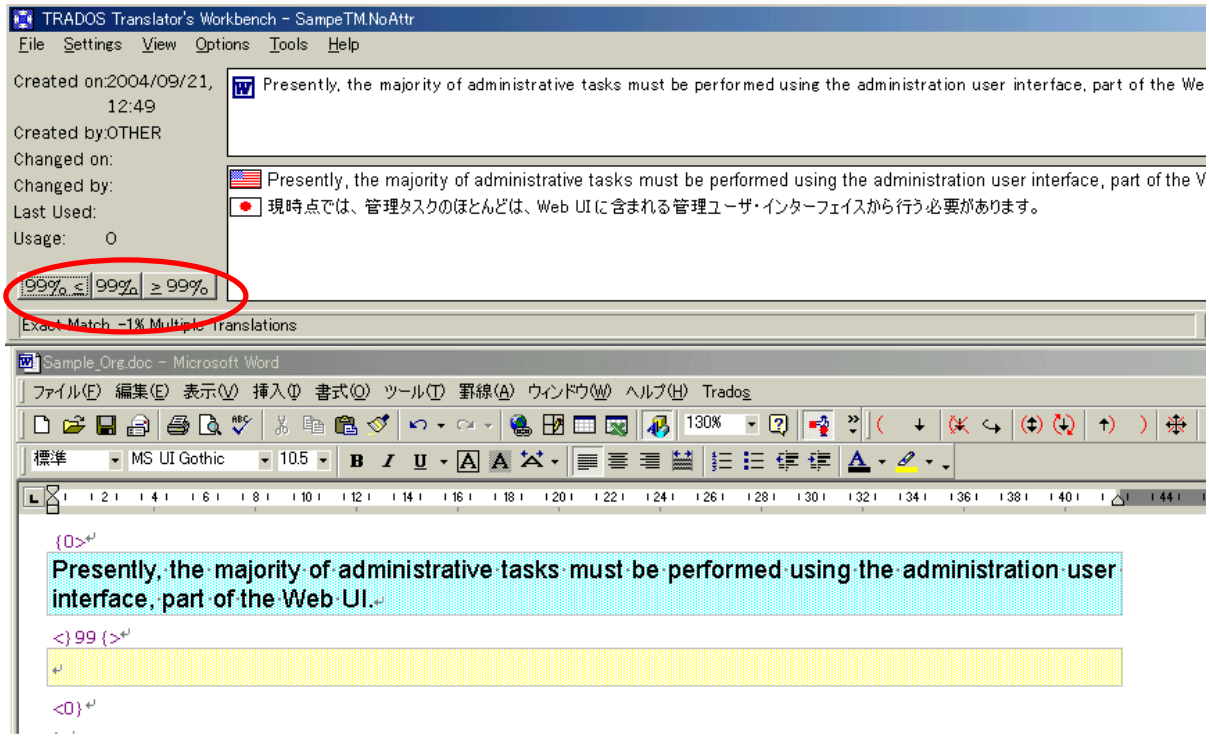
C). **翻訳者** C

{0>Presently, the majority of administrative tasks must be performed using the administration user interface, part of the Web UI.<}0{>現時点では、管理タスクのほとんどは、Web UI に含まれる管理ユーザー インターフェースから行う必要があります。<0}

D). **最終的に、カタカナ複合語には中黒 (・) を使用することになり、**user interface **は「ユーザ・インターフェイス」で統一することに決定**

{0>Presently, the majority of administrative tasks must be performed using the administration user interface, part of the Web UI.<}0{>現時点では、管理タスクのほとんどは、Web UI に含まれる管理ユーザ・インターフェイスから行う必要があります。<0}

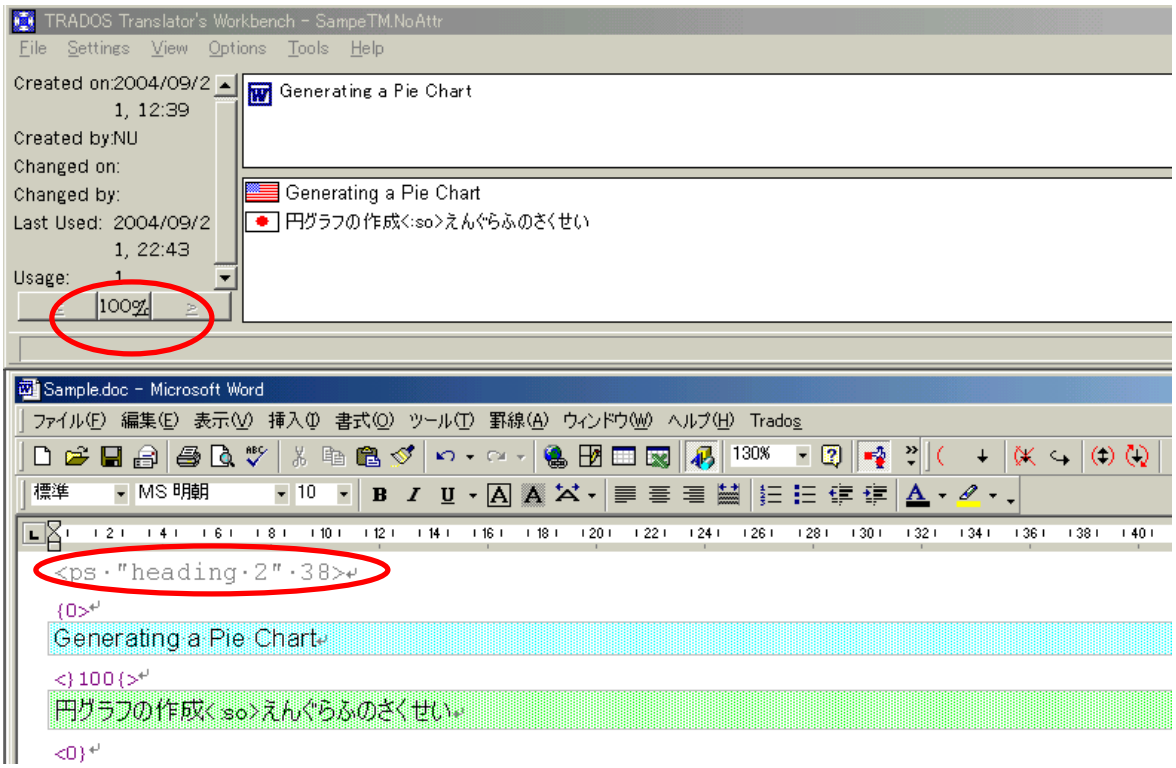
以上のような場合、各翻訳者のメモリを結合した結果、次のようになってしまいます。



実際には必要な訳文は 1 つだけなのに、同じ原文に 4 種類の訳文が存在し、いちいちボタンで候補を順にスクロールしなければならないなど、作業も煩雑になります。また、後になってから、この翻訳作業には関っていなかった第三者がこのメモリを使用することになったとき、どれが正しい訳なのか、メモリを見ただけではわかりません。

意図的な重複を意識せずに翻訳を進めた場合のメモリの例

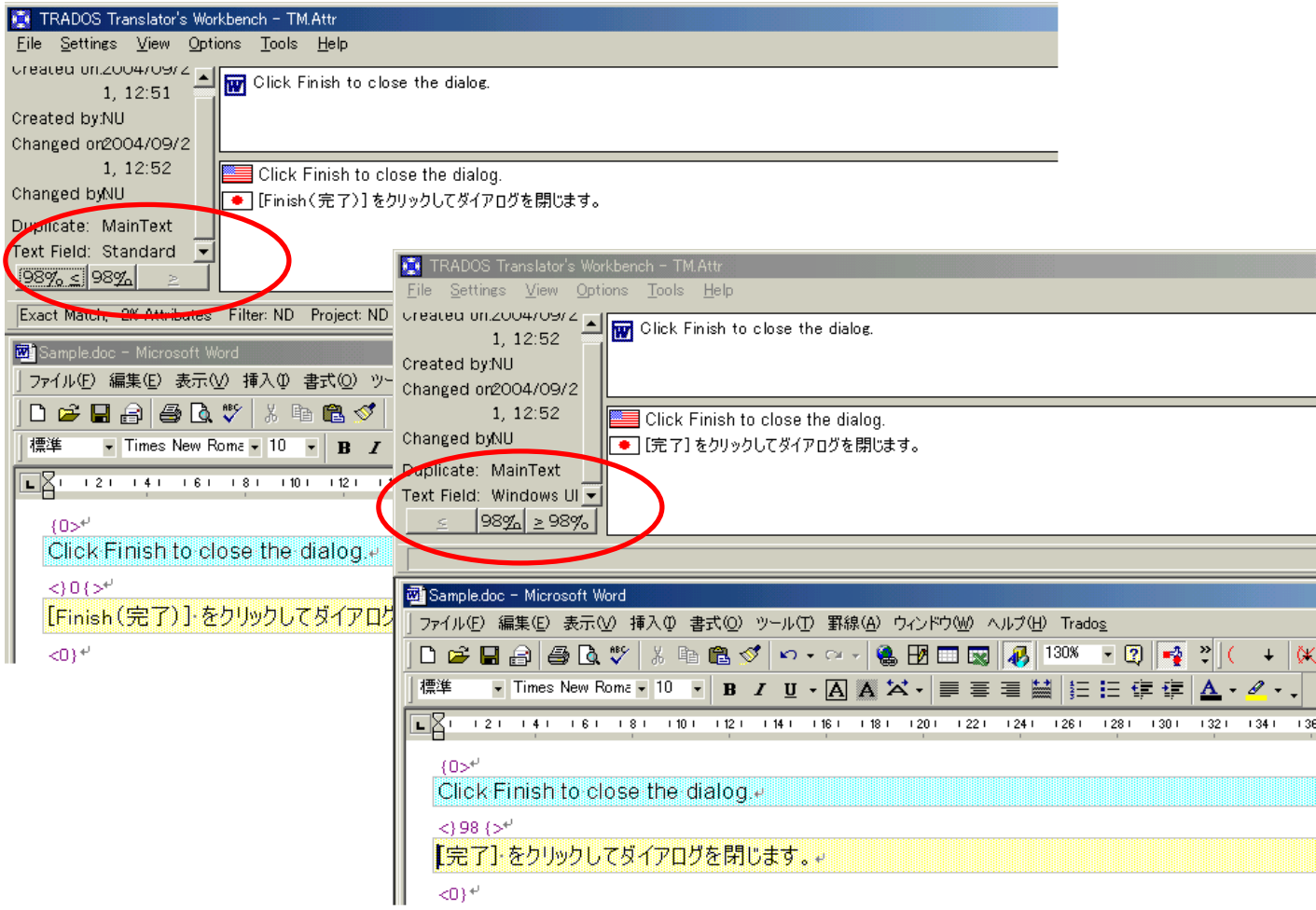
CASE 1. 同じ原文が索引マーカテキストと本文の両方で使われている (I-CASE2)



最初の翻訳時には状況に応じて 2 通りの訳出をしていたとしても、訳文をそのまま登録していたのでは、後からメモリに登録した訳文によって最初に入力した訳文が上書きされてしまいます。この状態で、後でこのメモリを再利用して自動翻訳した場合、上のように、原文は本文の章見出しであるのに、<:so>タグと読み仮名の付いた索引マーカ用の訳文が 100%一致として取得されてしまいます。索引 (<imk>と</imk>タグで囲まれたセグメント) 以外の場所で<:so>タグを使用すると、S-Tagger でタグ検証を行った際、エラーが出ます。逆に、索引マーカテキストに読み仮名のない訳文を取得すると、FrameMaker に変換した際に、索引が正しく生成されなくなります。

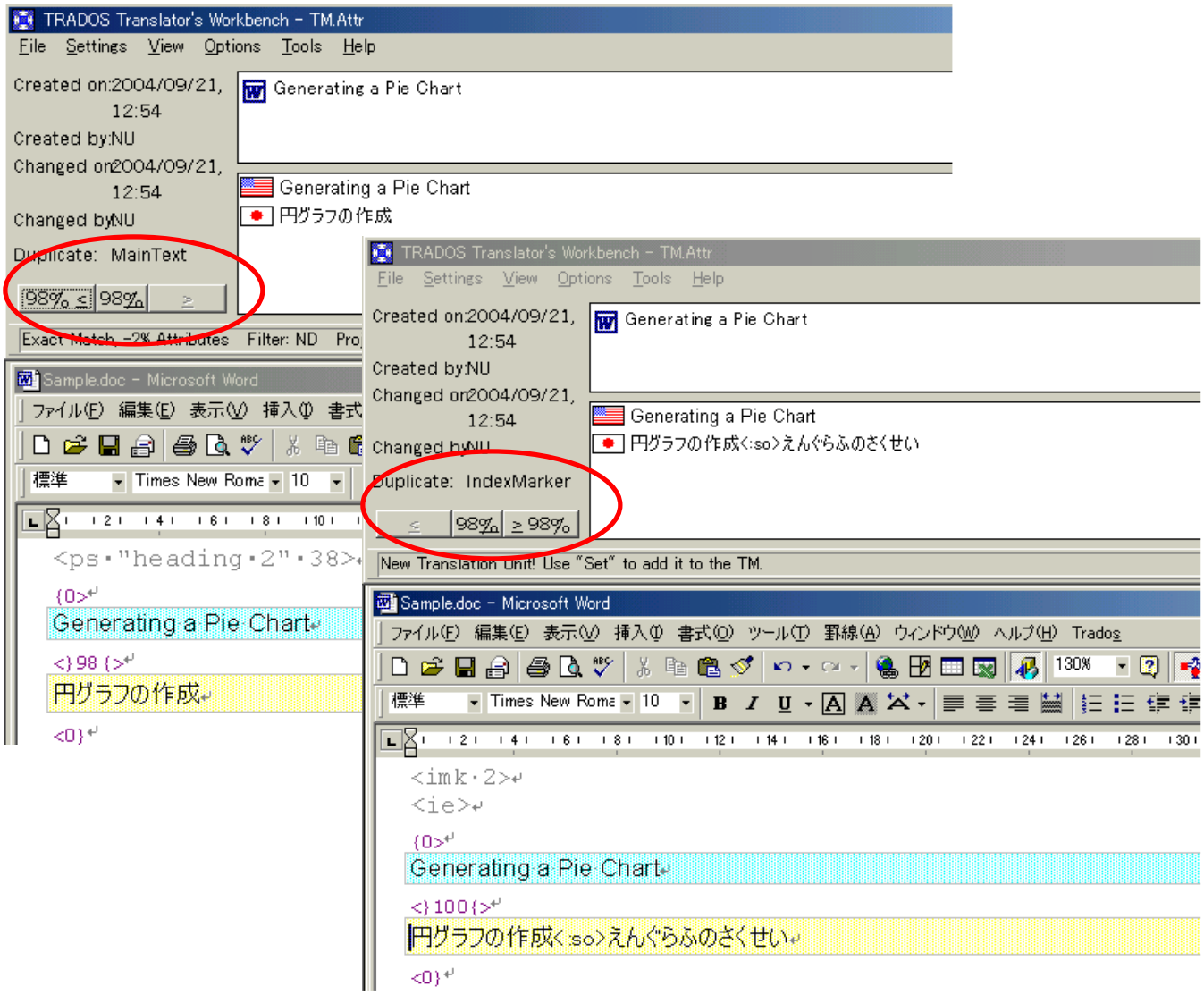
属性を活用した翻訳メモリの例

CASE 1. [English (日本語訳)] というスタイルが指定された UI と、完全に日本語化されている Windows UI が混在 (I-CASE1)



属性のペナルティ (2%) が付くことで訳文は 98%一致となり、自動翻訳をしたとしてもこの原文は処理対象にはなりません。必ず翻訳者の手を経なければ翻訳は行われないので、翻訳者がコンテキストを見ながら適切な訳文を手動で取得できます。また、属性を設定するとき、テキストフィールドにメモを入れておけば、後で第三者が見た場合でも、それぞれの訳文がどのような状況を意図しているものかがわかりやすくなります。

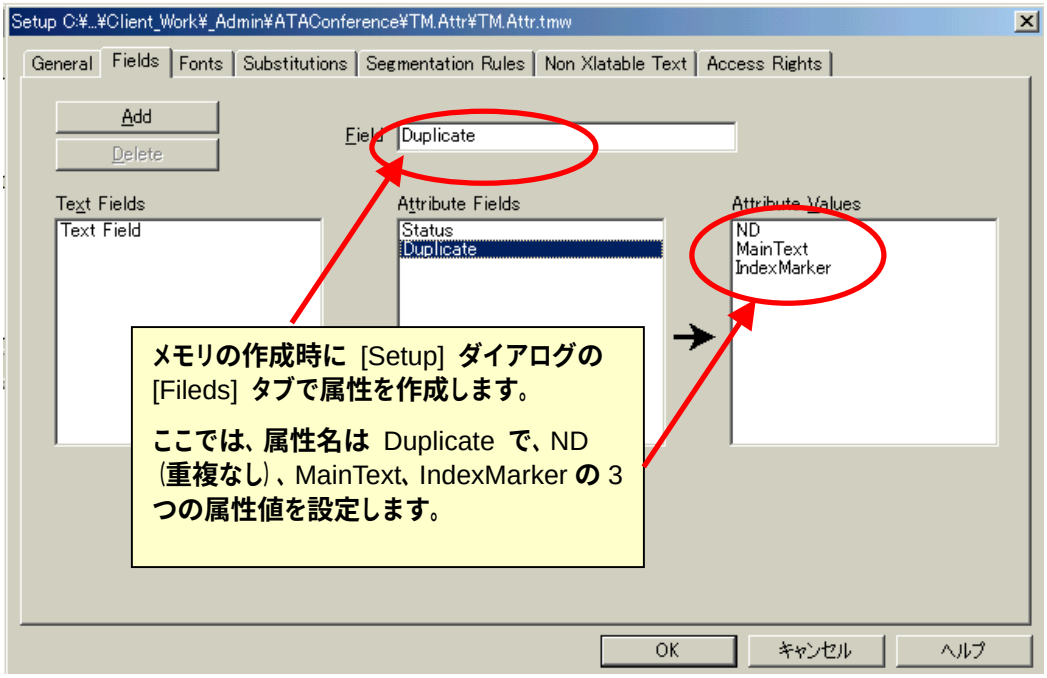
CASE 2. 同じ原文が索引マーカテキストと本文の両方で使われている (I-CASE2)



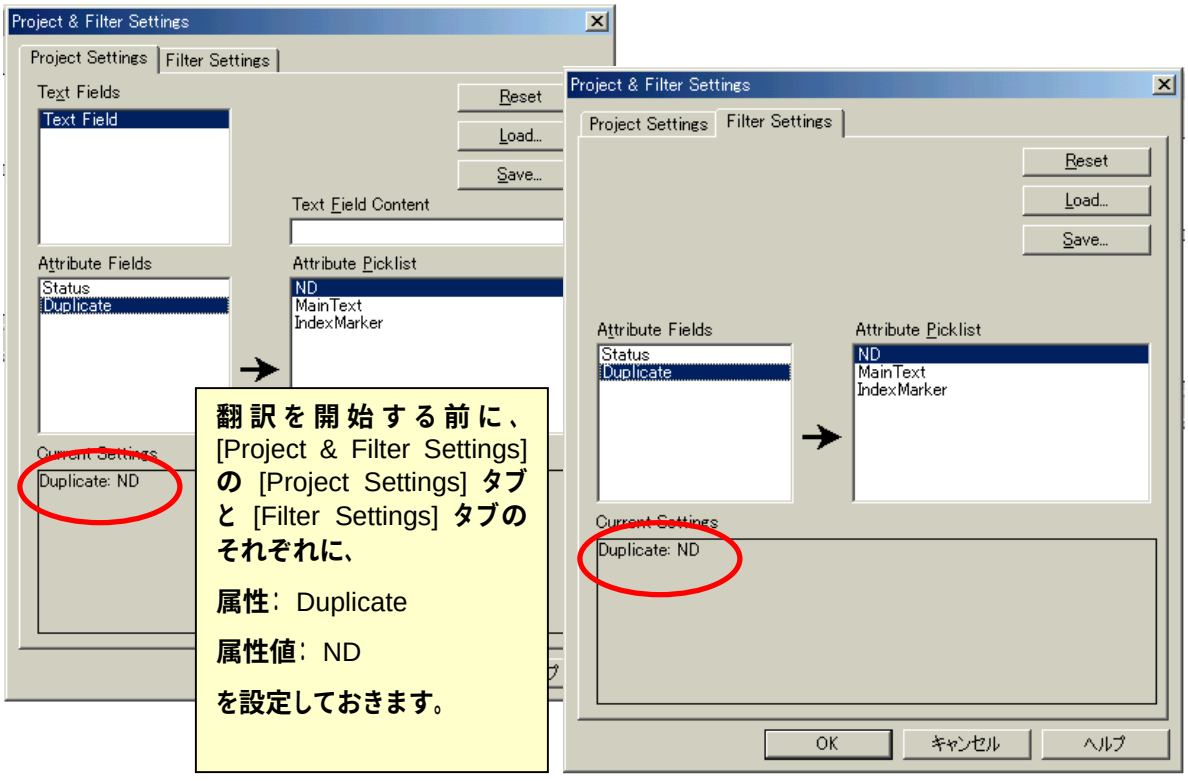
属性のペナルティ (2%) が付くことで訳文は 98%一致となり、自動翻訳をしたとしてもこの原文は処理対象にはなりません。必ず翻訳者の手を経なければ翻訳は行われないので、翻訳者がコンテキストを見ながら適切な訳文を手動で取得できます。また、第三者がこのメモリを使った場合でも、属性の名前から、それぞれの訳文がどのような状況を意図しているものかがわかりやすくなります。

付録：属性の設定方法

1. メモリ作成時に属性を設定



2. フィルタを設定



後は、翻訳をしながら、属性を設定する必要が生じた場合に、まずそのセグメントを開き、TW ウィンドウで対象の訳文を右クリックし、[Edit Translation Unit] を選択して属性をクリックで選択します。