2005－NL－166（6）
2005／3／10

Conceptualization of Program Integrated Information

加藤　直孝[1]　　有澤　誠[2]
(株)日本アイ・ビー・エム　ナショナル・ランゲージ・サポート[1]
慶應義塾大学　環境情報学部大学院　政策・メディア研究科[2]
katosan@jp.ibm.com[1]

概要: 本稿はソフトウェアのローカライゼーションにおける文字列の翻訳に関するものである。現在の多くのアプリケーション・システムはユーザーとのインターラクションを前提としており、ユーザーとの意思疎通のためにテキストを用いる。このテキスト情報のうちプログラムに統合した情報を Program Integrated Information (PII)とよぶ。PII はプログラム中に埋め込まず、プログラムとは分離した外部テキストファイル上に置く。これにより PII の翻訳者はプログラムのコードとは別に文字列を翻訳できる。ただし PII を外部テキストファイルにしても、PII の翻訳には多くの問題が残っている。プログラムの国際化に関して、従来は文字コードやファイルフォーマットと言った PII のプログラミングの側面のみを議論してきた。PII の翻訳や自然言語処理の側面は論じてこなかった。本稿は PII の自然言語、特に翻訳、の側面に焦点をあてる。まず、PII には「相」があることを指摘するとともに、PII のモデルを構成する「Feature」を導入する。そして、それら「相」や「Feature」が PII の文脈とどのような関係にあるかを提示することにより、PII の概念構築を行った。今後は構築した概念モデルを活用して、プログラム開発言語、プログラム開発環境、翻訳支援ツール、および自動翻訳プログラムがいかに協調して PII の翻訳の問題を解くかを議論できる。

キーワード: 翻訳、PII、Program Integrated Information、文字の外部化、地域化、国際化、モデル、リソースファイル

Conceptualization of Program Integrated Information

Naotaka Kato[1]　　Makoto Arisawa[2]
IBM Japan, Ltd. Translation Service Center.[1]
Keio University, Faculty of Environmental Information and Graduate School of Media and Governance[2]
katosan@jp.ibm.com[1]

Abstract: Most of the application systems require interaction with users. Such applications use text strings to communicate with users. Those strings are integrated in a software and are called “Program Integrated Information (PII).” This paper focuses translation of those text strings from the point of software localization. PII is separated into text files from application system programs. Although this separation enables translators to translate PII without referring the coding of the application system program, it does not solve all the problems that translators face when they translate PII. Computer scientists and engineers that were engaged in the design of PII have not discussed the natural language aspect of PII, but discussed only the programming aspect of PII such as character coding and PII file format. This paper introduces the PII concept of “So” and “Feature” to understand PII from both aspects. Then the conceptual model of PII is presented by relating the “PII Context” to the “So” and “Features.”

Keywords: translation, PII, Program Integrated Information, string externalization, localization, internationalization, model, resource file, PII architecture, MRI, Machine Readable Information, character coding, locale, initialization file

1. はじめに

1．1 プログラムの国際化と PII の翻訳の問題

国際化を必要とする多くのアプリケーションはユーザーとの意思疎通のためにテキストを用いる。このテキスト情報のうちプログラムに統合した情報を Program Integrated Information (PII)とよぶ。プログラムを国際化するには、この PII を何カ国語かに翻訳する必要がある。プログラムの開発組織では翻訳する可能性のある文字列は別のテキストリソースファイルに分離する。このテキストリソースファイルにはキーおよび分離したストリングを置く。そして、プログラム中にキーを埋め込み、キーの参照により適切な文字列を得る。IBM では、このテキストリソースファイルの翻訳作

業はプログラムの開発組織では行わず、翻訳の業務に特化した組織が行っている。加藤はこの PII の翻訳業務に従事しており、PII の翻訳および翻訳検証、翻訳の修正等のメインテナンスに伴うさまざまな問題に直面している。プログラムの国際化や地域化を扱ったほぼ全ての文献は、この PII の翻訳は翻訳の専門家が行うと記述している。いかにして PII を翻訳するかという記述が文献上にほとんどない。ところが、PII の翻訳ほど批判の的になるものはない。その理由は非常に簡単な語句の翻訳間違いが発生しているからである。そして多くのプログラミングを専門とする人たちは、アプリケーション・プログラム中に出現する不適切な翻訳を見つけては、翻訳の質を「これはひどい」と批判する。翻訳したプログラムを販売する人たちにいたっては、「まじめにやれ」と言わんばかりに翻訳を実行する組織に怒りをぶちまける。関係者の多くは PII の翻訳の問題の原因を翻訳者や翻訳組織の能力不足、努力不足に帰着する。しかしそれは誤りである。コンピュータ科学者やエンジニアにも半分責任がある。PII の翻訳には色々な問題がある。用語の統一と言った問題は翻訳者が注意して取り組む必要がある。しかし次に挙げるような、PII の翻訳が持った基本的な問題は翻訳者の努力だけでは問題は解けない。科学者、エンジニアの協力が必要である。

（1）翻訳時の問題 （文脈が不明）
（2）翻訳検証時の問題 （文字列の出所が不明）
（3）原典の更新に伴う問題 （どこに影響するか不明）
（4）誤訳修正に伴う問題 （付随修正箇所が不明）
（5）用語統一の問題 （意味を理解せず完全に統一するのは困難）

(1)翻訳時の問題: PII の翻訳者は本来ユーザーが使う GUI 上で翻訳を行うのではなく、文脈の存在しない外部テキスト上での翻訳を強いられる。外部化したテキストは比較的短い文章や語句が多く、GUI 上でのプログラムの流れを知らずに翻訳するのは、容易なことではない。外部テキストファイルを準備したプログラマーがたまにテキストファイル上にコメントを入れて翻訳者のために説明を加えている場合もある。しかし、そのケースはまれで、実際の GUI の流れを見ず適訳を決めるのは困難なことが多い。

(2)翻訳検証時の問題: 逆に GUI 上でプログラムの流れから誤訳を発見しても、いったいどこのテキストファイルのどのキーの文字列なのかわからない。この問題に対しては実際に有効な解決策を[1]で提示した。

(3)原典の更新に伴う問題: GUI のイメージをキャプチャーしたHTMLファイル等のマニュアルもPIIと関係する。PII を更新しても、いったいマニュアルのどこに影響するかわからない。

(4)誤訳修正に伴う問題: 誤訳の修正は(3)の PII の更新と同様の問題である。ただし影響は 1 言語にとどまる。PII のファイルにまたがった修正用語統一も容易ではない。

(5)用語統一の問題: 用語統一の問題は意味を理解しない単純な処理による翻訳用語不統一の検出プログラムがすでにある。そのプログラムは有益である。しかし、同一の原語を意味の違いにより意図的に翻訳を変えたものまで用語不統一エラーとして検出し問題も多い。

1．2 進歩を必要とする PII の翻訳環境

前節1．1でリストしたような問題はプログラムの規模が小さくて、PII の翻訳に十分な時間があり、PII の翻訳者とプログラマーが十分なコミュニケーションをとる余裕があれば、翻訳関係者の努力で解決することも可能である。しかし、プログラムが大きくて、色々な開発プロセスを同時並行で行い、限られた予算で短期間に PII を翻訳するには、もはや翻訳者および専門の翻訳組織の努力だけでは解決できない。これらの問題を解くには、ソフトウェアツールが必要でコンピュータ科学者やエンジニアの力が必要である。ところが、プログラム開発言語やプログラム開発環境はいずれも、翻訳の領域は自分の領域ではないかのように、翻訳する文字列を外部ファイルにした途端、話を終わらせる。一方、翻訳支援ツールは PII を十分考慮した翻訳のための機能を入れていない。

1．3 基礎概念の導入

本稿は、プログラム開発言語やプログラム開発環境、翻訳アシストツール、自動翻訳プログラム、翻訳者、といった色々なシステムが協調して PII の問題を解くための、PII の基礎概念を提供することを目的とする。そのために PII のモデルを構築することから始める。PII のモデルを使えば問題の全体を把握し、色々なシステムが協調して戦略的に問題を解くことができる。また、解決策が戦略的なものか、戦術的なものかの判断に、PII のモデルは役立つ。

2. PII の捉え方

2．1 PII の定義

IBM は PII を次のように定義している[2]。まず Machine Readable Information(MRI)を定義し、その中の一方を Program Integrated Information(PII)と定義する。

> 製品とその製品のユーザーとの間で交わす全ての言語および文化的影響を受ける情報はMachine Readable Information(MRI)とよぶ。
>
> MRI には次のものを含む。メッセージ・ダイアログ・ボックス、オンラインマニュアル、オーディオ出力、アニメーション、ウインドウ・ヘルプテキスト…、（以下細かい列挙が続く）。
>
> さらに MRI を 2 つのカテゴリーに分類する。プログラムに統合した情報(Program Integrated Information; PII)と統合しない情報 non-PII である。MRI のうち次の両方

の基準を満たすものが PII である。

（1）MRI 情報が製品が走っているときにのみアクセス可能な場合

（2）プログラムのユーザビリティーに対して、MRI 情報が基本的で必須である場合

例：ダイアログボックスとプログラムメニューは PII である。Non-PII 情報にはテュートリアルや製品資料がある。

本稿では PII をテキスト情報に限定して議論する。なお、これ以外の文献で同様の PII の定義を発見するのは困難である。

2．2 従来の PII の捉え方

ソフトウェアの国際化や地域化に関するほぼ全ての文献は PII をプログラミングの観点から位置づけている[3][4]。自然言語の観点から PII を位置づけている文献は見当たらない。例外的に PII を自然言語の観点から定義している文献として 2.1 節で採り上げた IBM のデザインガイドがある。ただしこの文献は経験を基にしたプログラマーのためのルールブックであり、PII に関する掘り下げた議論はまったくない。ほとんどの文献（たとえば[3]）は、PII に関して文字コードや英日と言った言語の切り替え等、プログラミングに関することを詳細に記述している。ところが、PII をユーザーとの意志疎通のためのコミュニケーションの道具としては捉えていない。意味を持った自然言語としては扱っていないのである。存在しても PII の翻訳に関するきわめて断片的な注意事項が数行あるだけである。

2．3 PII の多面性

上述したように PII には色々な側面がある。大きく捉えると 2 つの側面がある。そして従来は一方の側面にのみ焦点を当てており、もうひとつの側面を忘れていた。

PII を扱う立場による違いは次のようにまとめることができる。PII はユーザーとの意思疎通のためテキストである。しかし、扱う立場によって異なった見え方をする。プログラミングの視点からは、文字列型で定義する文字列変数の初期値である。その変数の初期値設定のファイルが PII テキストファイルとなる。プログラミングの立場からは、プログラム言語ごとに確立した手法に従った、初期値ファイルの切り替えにすぎない。さらにプログラム開発者は、その文字列が外延する意味をも決めている。ところが PII テキストファイルを翻訳する立場から見ると、PII テキストファイルは変数の初期値設定ファイルではなく、文脈がなく容易に意味を理解できない自然言語の断片となる。現状の PII 翻訳環境はその断片の翻訳を人間の翻訳者に強いている。1．1 節で述べたように翻訳者にとっては色々困難がある。一方、プログラムのユーザーは、PII を自然言語として捉えている。当然ユーザーがプログラムを使う時は言語の断片としてではなく、プログラムの流れの中で PII を理解できる。

2．4 コンピュータ科学の忘れ物

PII の翻訳を含むプログラム開発で、PII は二つの側面を持っていることになる。文字列変数の初期値としてとしての側面と自然言語としての側面である。PII を文字列変数の初期値の観点、すなわちプログラミングの観点、から解説した文献は多い。しかし PII を自然言語の観点から捉えた文献はほとんど見当たらない。PII に対する記述を PII の Feature とすると、変数の初期値であることも、自然言語であることもいずれも PII の Feature である。従来コンピュータ科学は、PII を複数の Feature を持ったモデルとして捉えてこなかった。PII の属性に関する議論もなかった。そのため、コンピュータ科学が PII に関し解決すべき問題を見過ごしてきている。PII にはどのような Feature があり、Feature 同士がどのように関係しているかを理解せずに、PII を扱うことはできないし、PII の翻訳者が抱える問題を合理的に解くこともできない。

どのようなモデルで PII を捉えればよいか。次の 3 章、4 章、5 章で詳しく述べる。

3．PII には相がある

3．1 文脈の有無と PII 相

図1の上半分は次の（1）（2）を示す。

（1）英語の PII ファイルを翻訳し日本語ファイルにするようす（図中央上部の2つのボックス）。

（2）それらの PII が GUI A、GUI B という2つの操作シナリオに従って GUI 上およびマニュアル上に現れるようす（図上部、左右の 6 個ずつのボックス）。

斜めや、横方向の矢印が PII の連鎖を示す。図1の下半分は、文脈の有無で図の上半分を分類し説明を加えている。“proposition”と “statement”を「命題」と翻訳し、“idea”を「発想」と「アイディア」と翻訳している。

図1の下半分に記述したように、PII が出現する世界は文脈のある世界と文脈のない世界に分かれる。ファイル上の PII には文脈は存在しない。プログラムとユーザーとのインターラクションが文脈を与え、PII は GUI 上に登場する。PII に相の変化が起きる。水が熱を加えると液体から気体に相変化するのと同様に、PII はインターラクションによって相を変える。PII は無味乾燥した文字コードの集まりから、意味を持った言葉に変わる。相が変わるとは、無味乾燥した文字コードが外延するものが見えるようになったり、見えなくなったりすることを言う。「液体の水」が「ファイル上の PII」にあたり、「気体の水」が「GUI 上の PII」と考える。「液体の水」は飲める。「気体の水」は吸える。これと同様に、観察者は「ファイル上の PII」と「GUI 上の PII」を違って受け止める。

人相、家相、手相、これらは全体がなければ部分のみでは意味を決定するのは困難である。目だけで人相はわ

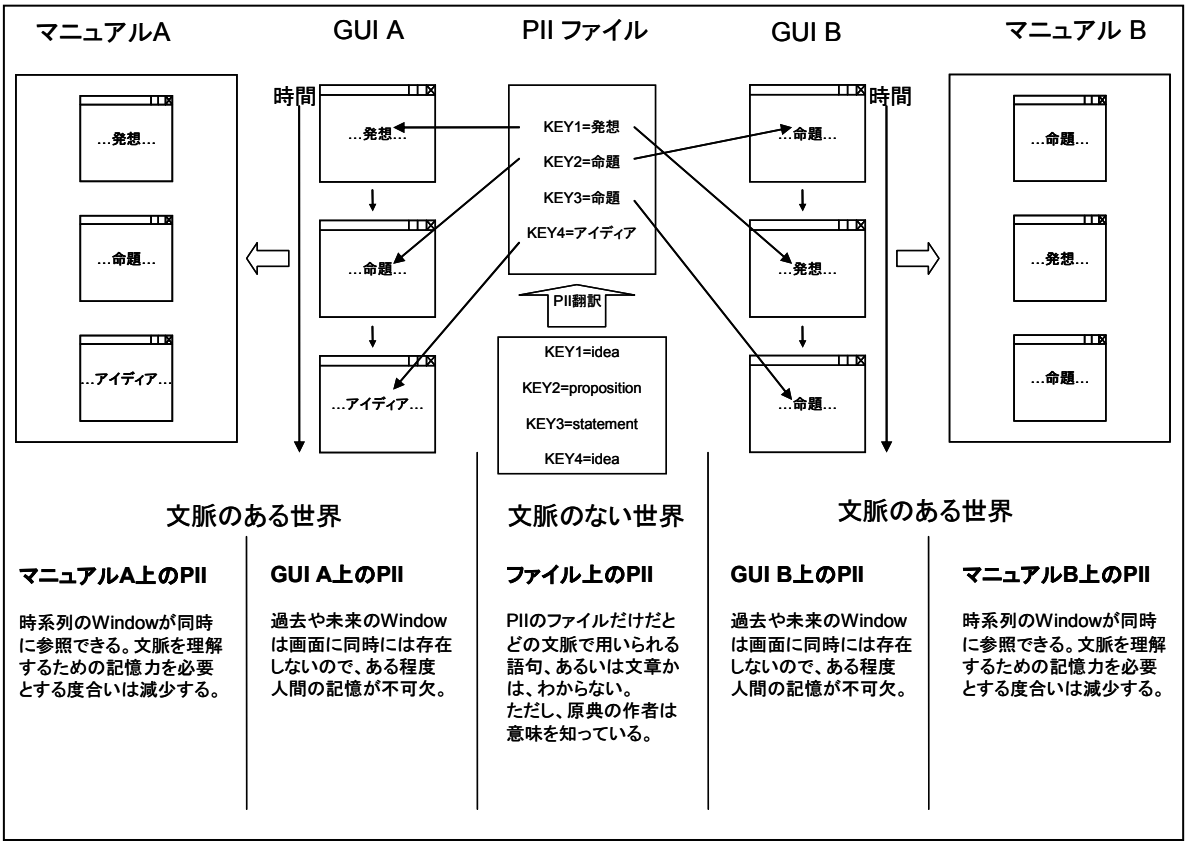

図 1　PII の連鎖および文脈

からない。寝室だけで家相はわからない。PII の相にも同様の全体と部分に対するニュアンスを含む。

「ファイル上の PII」は一定の範囲でしか意味を理解できない。しかし、インターラクションがあると、ファイル上の PII は文脈のあるGUI上の世界へ転移する。PIIは文脈の壁を越える。そして、「GUI 上の PII」が出現し、正確に PII の意味を理解できるようになる。GUI 上では時系列で文脈を観察する必要があり、マニュアル上では空間的に文脈を観察できる。

PII はもともとファイル上に置いたときからある限定した意味をもっている。その意味は原典の作者のみが知っている。ただそれは、観察者がファイル上で観察しても、その限定した意味を知ることはできない。PII の「相」が変わらない限り観察できない。作者以外の者は、ファイル上でどんなに「idea」や「発想」という文字列を観察したところで、意味を限定することはできない。観察者には文脈が必要であり、文脈を生むのはプログラムとユーザーとのインターラクションだからである。PII の中身は同じだが、観察するにはその「相」が変わっている必要がある。

おもしろいことには、PIIの作者は全ての文脈を知っているわけではない。文脈を生み出しているのは PII の作者ではないからである。文脈が PII の意味を限定したのではなく、PII はもともと限定した意味をもっている。たとえば、KEY1 の「idea」と KEY4 の「idea」は作者にとっては元々意味が異なる。GUI 上で出現する場所も異なる。もし意味の異なる「idea」を同じキーに与えると、翻訳時に困難に直面する。KEY1 を「発想」、KEY4 を「アイディア」といったように訳せなくなってしまうからである。PII の意味は元々限定しておかなければならないのである。

3．1．1 PII の作者と観察者

上記3．1節ではPIIの原典作者と観察者を分けてPIIを捉えた。言語全般においても当てはまるかどうかは断言できない。しかし、少なくとも PII に関しては、次のような仮説を立てると分かりやすい。

語句の意味を決めるのは、作者と観察者では異なる。

(1) 作者は過去の多くの文例パターンから語句の意味を決定し、自分の PII ファイル中で使用する。意味が先、文脈は後。

(2) 観察者は、文脈から語句の意味を決定する。語句は大雑把な意味の範囲を示すにとどまる。文脈が先、意味は後。ただし、決定するときその語句が観察者が頭の中にもつ過去の文例パターン中の語句と同種かどうかの確認をする。

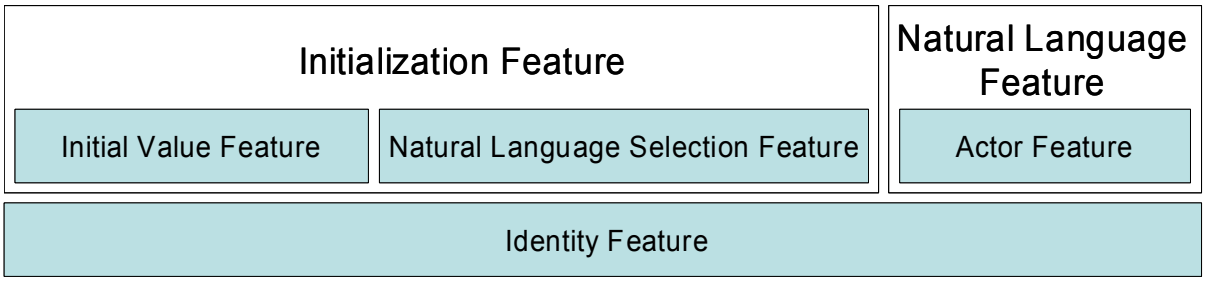

図 2　PII Features

たとえば、上記(1)に関して、文脈でそれぞれの語句の意味が変わるような語句を PII には与えてはならない。PII 翻訳不能となる。PII は PII が独立して1対1の翻訳が可能なように PII のユニットを決めていないといけない。ひとつの PII に2つの訳は付けられないからである。上記(2)に関しては次のようなことがある。観察者(ユーザー)が観察者の頭の中にある過去の文例パターンから特定の語句が外延しうると思うものと、実際に文脈から作者が原語で外延しただろうと思うものが異なるとき、観察者は翻訳が悪いと苦情を言う。たとえば、英語原典 “Input value is null.”を「入力値が null です」と翻訳すると苦情が来て、「入力値が無効」にせよと言う。null を「空」とするか「無効」(invalid)とするか迷う。翻訳は結局、「値が入力されていません。」とした。

3．2 PII の連鎖

本節では、図1の上半分を説明する。文脈の観点からは PII には文脈の壁がある。一方一つひとつの PII には、相や文脈の壁を超えた連鎖が存在する。

図1の上半分に PII の連鎖を示す。PII ファイルとあるのは、外部化した文字列がテキストファイル上に存在している状態である。たとえば、1 つのキー(KEY1)に対して原語「idea」がある。翻訳により、別ファイルを作成し、KEY1 に対しては「発想」を与えている。PC の操作が異なれば、出力する GUI も異なる。図1では 2 つの PC の操作を左右に示した。GUI A と GUI B である。操作の順に上から 3 つの Window を表示したことを示す。GUI A では「…発想…」は最初の Window で出現し、GUI B では 2 番目の Window で「…発想…」が出現している。

多くの場合アプリケーション・プログラムにはマニュアルが存在し、GUI をイメージ・キャプチャーして載せている。マニュアルとは html フォーマットのファイルや紙に製本したものを指す。マニュアルは時系列に出力する Window を順に掲載することにより GUI 全体の流れを示すことが多い。各イメージに対して GUI 中の文章や語句を引用して、GUI について説明する。たとえば、GUI 上に出力する「発想」という語句は、イメージとしてマニュアル上に出現すると同時にマニュアル中の GUI を説明する文章にも出現する。PC 操作では、一時に 1 つの Window しか見ることができなくても、マニュアル上だと、同時に複数の Window が参照でき、GUI の流れを容易に把握できる。「idea」という 1 つの KEY1 の値は、日本語ファイル上の「…発想…」、GUI A 上の「…発想…」、マニュアル A 上の「…発想…」、GUI B 上の「…発想…」、マニュアル B 上の「…発想…」と全てつながっている。図には示さないけれども、アプリケーション・システムに他の言語、たとえばドイツ語のサポートがあればそのつながりはさらに大きなものとなる。なお、キーの値は単語のみならず、語句や文章を含む。

たとえば、次のような状況がありうる。PII のある文字列を修正したとする。ただし、開発元の英語原典の PII を想定する。PII の修正は外部テキストファイル上で行う。すぐに困った問題が起きる。実際の GUI 上の表示とすでにできているマニュアル上との表示が食い違ってしまう。英語原典に関してはマニュアルの問題で済む。しかし、PII の翻訳がある場合は、翻訳した文字列を修正しない限り、原典英語以外の GUI には何の修正も行われない。PII およびその翻訳では、PII が関係するもの全体に手を打たなければ、本来 PII が持つ連鎖は切れてしまう。文字列変数の初期値(ファイル相)としての PII は GUI 上の自然言語(GUI 相、マニュアル相)としての PII と結びついている。

4 PII Features

本章では、PII の世界を記述している Feature にどのようなものがあるかを明らかにすることにより、PII のモデルを構築する。また Feature と文脈および PII 相との関係を明確にし、PII の基礎概念を構築する。図 2 に PII の Feature およびそれらの関係を示す。

PII の Feature は大きく 3 つに分かれる。

(1) Initialization Feature
(2) Natural Language Feature
(3) Identity Feature

(1)を 2 つに分けると次の 4 つとなる。

(1) Initial Value Feature
(2) Natural Language Selection Feature (NLSF)
(3) Actor Feature
(4) Identity Feature

2 番目の Feature を NLSF と略す。

次にこれらの Feature を説明し、文脈や PII 相との関係を説明する。

4．1 Initial Value Feature、NLSF

プログラムが GUI 上に出力する文字列変数の初期化に関する Feature を Initialization Feature と命名する。プログラムミングの観点から眺めた PII の Feature と言ってよい。具体的には翻訳可能性のあるプログラム中の文字列の外部化に関するさまざまな機能によって Feature を構成している。たとえば、文字コード、PII のファイルフォーマット、ロケールの設定等の機能で構成している。従来のプログラムの国際化、地域化に関する文献は、この Feature に関する解説を行っている。この Feature は基本的には 2 つの Feature に分かれる。1 つは、変数の初期値を決める Feature であり、もう 1 つはプログラムの言語環境に合わせて、初期値を切り換える Feature である。前者を Initial Value Feature と命名し、後者を Natural Language Selection Feature（NLSF）と命名する。前者は、*.ini ファイルや*.properties ファイルに見られるようにプロパティーなどの初期値設定の一般的な Feature である。ただし、PII では初期値は文字列と決まっている。後者は、言語環境あるいはプログラムのロケール設定に従って、初期化ファイルを選択する機能で構成している。ファイル名（e.g. filename_ja.properties)を活用して環境にあった文字列ファイルで初期化する機能などがそれにあたる。

4．2 Actor Feature、Identity Feature

4．2．1 Actor Feature

PII を単に文字列変数に与える値に過ぎないと考えるのでは、PII の半分の側面しか見ていないことになる。もう 1 つの側面は PII をプログラムとユーザーとの意思疎通のための道具と捉える見方である。PII の自然言語の側面である。本稿では、この PII の自然言語に関する Feature を Natural Language Feature と命名する。本稿では、PII の Natural Language Feature のうちプログラムとユーザーとの意思疎通のための Feature を Actor Feature と命名する。

Actor Feature という名前は、PII を比喩的に演劇における俳優（Actor）として捉えると分りやすいので用いた。PII は文脈のないテキストファイル上の世界（舞台裏）で出番を待っている。プログラムとユーザーとのインターラクションが PII に文脈を与え、そのとき PII は GUI 上に現れその役割を果たす。同じ、語句や文章がファイル中にあるときには、文脈のない世界に存在し、PII は「ファイル相」を持つ。GUI の中に現れるときには、文脈のある世界に存在し、PII は「GUI 相」をもつ。マニュアル上では、GUI 上で時系列に存在した PII を空間的に見ることができる。この PII は「マニュアル相」を持つ。PII という俳優(Actor)がそれぞれの舞台で役割を果たしているという意味で、この PII の Feature を PII Actor Feature とした。

4．2．2 Identity Feature

4．2．2．1 PII の語句や文章は「着ぐるみ」

PII を比喩的に俳優が「着ぐるみ」を着て演技を行うと考えると分りやすい。PII がテキストファイル上にあるときは、俳優が着ぐるみを脱いで楽屋にいるときに相当する。テキストファイル上のキーが俳優であり、初期値としての語句や文章は「着ぐるみ」にあたる。俳優（キー）は出番がくると「着ぐるみ」（語句）を着て舞台（GUI）に出演する。観客（ユーザー）が見るのは舞台（GUI）と「着ぐるみ」（語句、文）であり、俳優（キー）は見えない。

図1の上半分を演劇の比喩で説明すると、翻訳前が米国公演のそれぞれの俳優（キー）と「着ぐるみ」の関係を示す（初期値としての文字コード）。翻訳後が日本公演のそれぞれの俳優と「着ぐるみ」の関係を示す。KEY1 が俳優1、KEY2 が俳優2、以下同様である。米国公演では俳優1は「idea」の着ぐるみ、俳優2は「proposition」の着ぐるみを衣装とする。日本公演ではそれぞれ「発想」「命題」の着ぐるみとなる。たとえば、舞台に出ると、俳優2、俳優3と言った俳優自身は見えない。俳優2、俳優3に関しては、日本公演では「命題」という同じデザインの着ぐるみで舞台に登場する。しかしそれぞれの俳優は異なった演技をし、それぞれの「着ぐるみ」は異なった俳優であることが観客には理解できる（はずである）。そうでなければ、日本公演でも、「着ぐるみ」に何らかの違いが必要である。もとの米国公演では、「着ぐるみ」も異なっている。「着ぐるみ」の比喩を使わず日本語の KEY2、KEY3 の GUI 上での説明をすると、同じ文字コードのものが見えていても、GUI 上ではその文字コードが外延するものが異なるということである。ここでは、双子のドラマを思い浮かべてもらった方がよいかもしれない。

4．2．2．2 PII には Identity がある

ユーザーは文字コードがもともと意味するものと文脈により具体的な意味を理解する。ファイル上の初期値としての文字コードにも意味はある。しかし意味する範囲が広く、それが外延するものははっきりしない。同じ文字コードを GUI 上で見ると外延するものがはっきり分る。ここで重要なことは、一度外延するものが決まると、ユーザーは見えないはずのキーを突き止めたのと同じで、この「命題」は KEY2 の「命題」なのか KEY3 の「命題」なのかと言う判断をするようになる。ユーザーは「…命題…」の Identity を突き止めたことになる。KEY2 という Identity と KEY3 という Identity である。PII テキストファイル中の代入文 1 つ 1 つに Identity があるので、この PII の Feature を Identity Feature と命名する。

図2で示すように、この Identity Feature はすべての Feature にまたがって存在している。3．2節で連鎖が存在するのは、この Identity Feature が相をのりこえて存在する

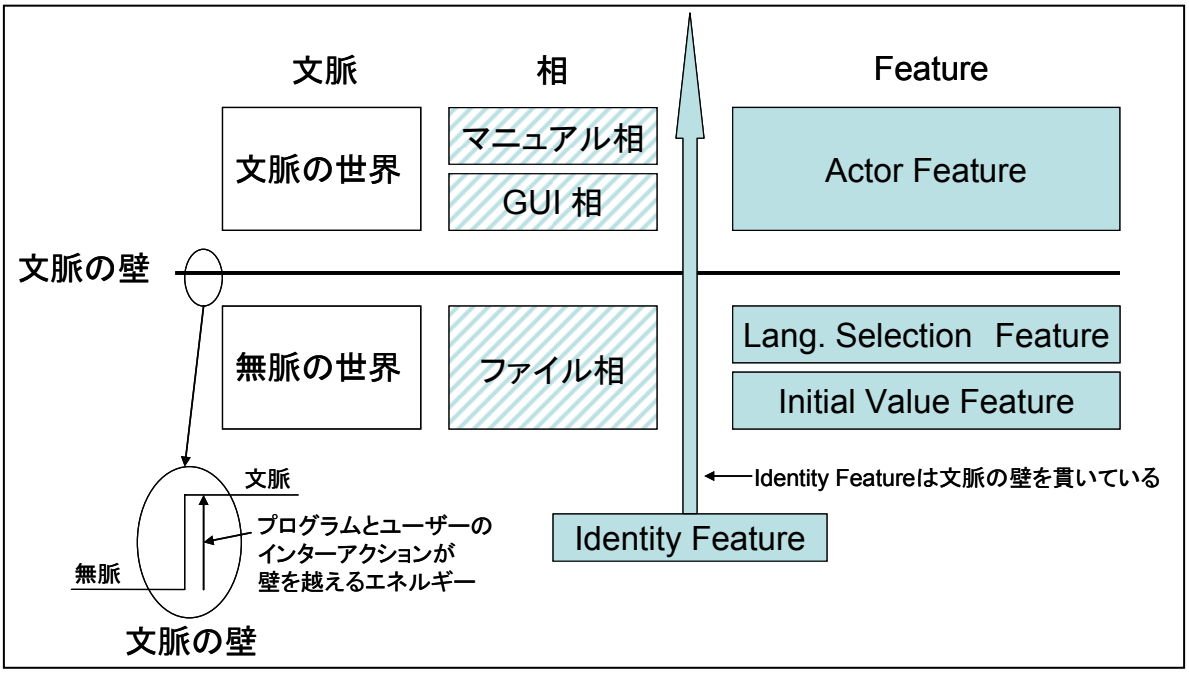

図 3　PII の概念：　文脈、相、および PII Features

からである。Identity Feature は5章で説明するように、文脈を含めた PII の概念を理解するための鍵となる Feature である。

4．2．2．3　PIIの Identity Feature と IBM の経験則

IBM のデザインガイドは経験則に基づく IBM の国際化のためのルールブックである。その中に Identity Feature と関係のあるルール（Rule A5）がある。本項ではそのルールを紹介する。

Rule A5 – メッセージのトラック（追跡）とダイアログボックス

Rule A5: メッセージやダイアログボックスと言った MRI をトラック（追跡）する方法を翻訳者に提供すること。

Guideline A5-1: アプリケーションまたは Web サイトでの変更を識別する仕組みを持たせること。また、その変更情報を翻訳または地域化のプロセスに渡すこと。

例として、翻訳者が Manual を翻訳するときには、唯一無二の識別子(Unique Identifier)によって、メッセージを特定できるようにすること、と明記している。PII に限定して考えると、メッセージが特定できるとは、GUI 上のメッセージ（語句や文章）がどの PII のメッセージであるか特定できるようにするという意味である。本稿の PII の概念で述べると、ファイル相、GUI 相、Manual 相とそれぞれ相の異なる PII の連鎖がわかるようにせよ、ということである。Identity Feature が翻訳者に見えるようにせよ、ということである。1980 年代の IBM の大型ホスト端末（3270 端末）のフルスクリーンの右上にユーザーにとっては意味のない ID が入っていた。フルスクリーン画面という MRI に対する Rule 5A の実装である。GUI ベースのアプリケーションに移行したとき、このルールは忘れられてしまった。モデルがなく、ルールの意味の理解が困難だったのかもしれない。

5　PII の概念

5．1　PII における文脈と無脈

本節では、PII の Feature や相の観点から、PII の文脈を位置づけることで、PII の基礎となる概念を示す。

文脈に関して PII の世界では極めて特徴的なことがある。PII が文脈のある世界とない世界に存在し、それぞれの PII が Identity を失わないことである。本や新聞のように人間のみが作り出した文脈では、そこに登場する語句や文章は元どこにあったのかを議論するのは容易ではない。PII の場合は文脈のない世界における存在場所が明確である。テキストファイル上でのキーとセットになった存在である。本稿では文脈のない世界を無脈の世界とよぶことにする。

図3に PII の Feature や PII の相と文脈に関する関係を示した。文脈の世界と無脈の世界の間には壁がある。これを「文脈の壁」と命名する。PII は、テキストファイル上にいるときには無脈の世界に存在し、GUI 上に登場するときには文脈のある世界に存在する。Initialization Feature を構成する機能は無脈の世界の機能である。一方 Actor Feature を構成する機能は文脈の世界の機能である。図3にあるように、文脈の壁は PII の世界を2分している。

PII は2分した世界に存在する。しかし、PII には Identity Feature があり、この2分した世界を1つにまとめている。PII

はテキストファイル上でファイル相をもつ。そしてIdentityをもっている。(Identity と一意に対応しているのはフォルダー名を含むファイル名とキーである。)ファイル相の PII を GUI 相の PII に変えるにはエネルギーが必要である。外延するものが見えない状態から外延するものが見える状態に変えるエネルギーである。プログラムとユーザーとのインターラクションがそのエネルギーを与える。プログラムとユーザーとのインターラクション(Interaction)のエネルギーがPIIをファイル相からGUI相に活性化(Activate)する。文脈の壁を突き破り Inactive な状態から Active な状態に変化する。そして Actor Feature が有効になる。

5．2 PII と対訳表(辞書)の関係

「無脈の世界」のもうひとつ手前には「対訳表」が存在する。ここで PII と対訳表(辞書)の関係を説明する。

「IBM 情報処理用語英和対訳集」という英日の対訳集がある。言葉の定義が書いているわけではないので、辞書ではないけれど、PII を翻訳する組織では「辞書」と呼んでいる。単語や語句を含み、文は含まない。この対訳集は PII の翻訳がもとのデータになっている。対訳表の世界とは「無脈」の世界の言葉の部分集合と考えうる。しかし実際には、「文脈」の世界で決定した訳語を、対訳リストに加えている。

PII と対訳集との決定的な差は、PII には文脈の世界があり、Identity をもっている。一方、対訳集には文脈の世界が存在せず、Identity もない。対訳集と PII を分けているものは PII の Feature である。演劇の比喩を使うと、対訳集は「着ぐるみ」の倉庫と理解してよい。「着ぐるみ」に PII のファイル上でキーを与えたとき初めて Identity は生まれる。

ある英語の PII に適切な訳語が無かったとする。倉庫に役に合った「着ぐるみ」がない状態である。そのときには新しい訳語を作る必要がある。PII の翻訳では、実際に新しい訳語を作ることもある。対訳表を更新するときに、この訳語は新たに対訳表に登場する。PII から Identity Feature、Actor Feature、Initialization Feature を取り去りリストに加えることに他ならない。新しい「着ぐるみ」の倉庫への追加である。

PII 翻訳の世界では、語句は「対訳表」→「無脈の世界」→「文脈の世界」と言う流れをたどる。対訳表の作成は「文脈の世界」→「無脈の世界」→「対訳表」の流れをたどる。

6．PII の分断から統合へ

プログラム言語やプログラム開発環境は PII の翻訳のために多くのなすべき事がある。逆に、自然言語処理の分野は、プログラム言語やプログラム開発環境と協調して PII の翻訳を行う必要がある。プログラムとユーザーとのインターラクションで文脈は生まれる。そのインターラクションの素となる情報はプログラムの中にある。自然言語処理のためにそれらを活用しない手はない。

PII の翻訳者は PII がどのような GUI で使われるのか、テキストファイル上の PII を翻訳するときに見たい。しかし現状ではそれは容易ではない。PII の翻訳検証では GUI がどこのファイルのどのキーの PII かを知りたい。プログラム開発環境の協力があれば[1]より更に便利なシステムができる。

翻訳支援ツールは、多様な翻訳の中の 1 つの種類として PII の翻訳を行っている。本稿で述べたような PII の特徴を十分生かした翻訳支援はできていない。

PII の翻訳は、関係する分野の人々が垣根をこえて研究を行い協力すれば、大きな進歩が期待できる分野である。

7．おわりに

[1]の研究会の発表を[5]で佐藤理史先生に取り上げていただいた。その内容に触発され、本稿は生まれた。[7]の辞書に対する見方と本稿の対訳表に対する捉え方には共通のものがある。また、[7]にマンションの部屋割りの例がある。相の観点からは区分により家相が生まれる。

PII は Identity を持ち、Actor としての Feature も持っている。あたかも人間のようである。何らかの使命を持ってこの世に生まれ、そして消滅する。従って履歴や属性を持つ。これらについては稿を改めて記述したい。

謝辞

[1]の研究会では、京都大学の佐藤理史先生はじめ色々な方から示唆と励ましをいただいた。本稿ができたのは皆様のおかげである。ここに謹んで感謝の意を表する。

参考文献

[1]加藤直孝，有澤誠，“翻訳検証テストのためのストリングリソース ID”，情報処理学会研究報告， 2004-NL-162，pp. 95-102 (2004)

[2]IBM, Designing Internationalized Products, National Language Design Guide Volume1, Fifth Edition, February 2004

[3]Andrew Deitsch and David Czarnecki, Java Internationalization, O'REILLY, 2001

[4]Nadine Kano, Developing International Software for Windows95 and Windows NT, Microsoft Press, 1995

[5]佐藤理史，“「言い換え」特集号編集後記”，自然言語処理，Volume11，Number4，pp. 209-210, 2004/10

[6]佐藤理史，“境界認定の提案: (1) コンセプトと実現法”，情報処理学会研究報告， 2004-NL-164, pp. 25-32 (2004)

[7]佐藤理史，“境界認定の提案: (2) 背景と思想”，情報処理学会研究報告, 2004-NL-164, pp. 33-40 (2004)

[8]Java Tutorial: Internationalization, http://java.sun.com/docs/books/tutorial/i18n/intro/index.html, June 2004

[9]Testing your internationalized Eclipse plug-in http://www-106.ibm.com/developerworks/opensource/library/os-i18n2/, July 2002