【様式4】

テーマ名：実世界志向プログラミングのための分散人力処理環境の開発
申請者名：馬場匠見

【提案テーマ詳細説明】

1. 提案内容

1.1. 概要

本提案では、プログラム内において、「人」をセンサーやアクチュエータ、計算機と同じように扱える仕組みを実装する。

クラウドソーシングやヒューマンコンピュテーションなどの概念が一般化していくにつれ、プログラムの中で「人」への命令についての記述を行うことが増えている。しかし、既存のクラウドソーシングなどの仕組みでは、身近な人や自分の周りの実世界環境や計算機環境とのインタラクションを記述することは考慮されていない。

そこで、プログラムからセンサ・アクチュエータや計算機を扱うことと同じように、「人」を扱えるような仕組みを実装する。具体的には、プログラムから人に対し命令を送れるライブラリと、プログラムからの命令を受け取り、人が処理した結果をプログラムに返すようなアプリケーションを実装する。この仕組みによって、プログラム上で人をセンサ・アクチュエータや計算機と同等に扱うことができるようになる。また、分散処理の仕組みを取り入れることによって、多数の人に命令を分散したり、人とセンサやアクチュエータ、計算機を垣根を取り払った実世界志向プログラミングのための環境を実現する。

プログラム上で、人とセンサ・アクチュエータや計算機が同じようなコードで動作し、お互いに通信をしあえるような環境は実世界志向プログラミングの新しい形であると言える。

1.2. 背景

近年、プログラムにおける人の扱い方が変化している。今まではプログラム内に、「人」について記述することはあまりなかった。プログラムはプログラムで簡潔しており、「人」という曖昧な要素は排除されてきていた。しかし、最近になって「人」というリソースをうまく活用するためにプログラム内にて、「人」について記述することが増加してきている。コンピュータから人に何か行動させるようなことが増えているのだ。Amazon Mechanical Turkを始めとするクラウドソーシングサービスの登場によって、プログラムから人に対して仕事を割り振るような仕組みはビジネスにおいて使われるようになった。また、人をセンサーとして利用することで、よりコンテキストを纏ったデータであったり、センサーでは取りにくいデータをセンシングするための試みも行われている。人リソースをプログラム内で扱おうという需要は間違いなく増えてきている。

プログラム内で人リソースを扱おうとする理由としては、まだまだコンピュータよりも人間のほうが得意なことが多いということが挙げられる。人の感性に関わるような事柄は、コンピュータでは判断しにくい。例えば、「写真Aと写真Bを比較した時にどっちのほうが見栄えが良いですか？」といった問題の場合、コンピュータが判断することはとても難しい。人間の場合ならば、見栄えが良いなどの感性的な判断はすぐにできる。何か実世界で行動を起こさなくてはいけないことも、コンピュータは不得意だ。コンピュータが実世界で行動するためには、アクチュエータなりが必要となってくるが、自由度は高くはない。「料理をするのでほうれん草を買ってきて」という命令の場合、コンピュータのみで実現することは難しい。人間の場合ならば、すぐにできることだ。

コンピュータは確かに有能であるが、できないことも多い。コンピュータが得意なことはコンピュータが、人が得意なことは人がやるといった住み分けが求められる。当たり前のことではあるが、これは実生活には浸透してはいない。AmazonMechanicalTurk等のクラウドソーシングサービスはコンピュータと人の住み分けの良い例であると言えるが、対象となる人は世界中に分散されており、家族を対象にしたり、ある特定の場所の近くにいる人、といったことはできない。コンピュータと人の住み分けはビジネス用途などに限られてしまっている。

ユビキタスコンピューティングの世界でも人をセンサとして扱うような研究事例も増えており、様々な分野において、人をセンサ・アクチュエータや計算機として扱えるような仕組みが求められている。将来的には、現状のセンサ・アクチュエータと計算機の関係に、人が大きく関わるようなモデルになっていくのではないかと考えられる。

1.3. 提案内容

本提案では、プログラム内における「人」をセンサ・アクチュエータや計算機と対等に通信し合えるような環境の実装を実現させる。この環境において「人」は、1つのプロセスのようなモノとして認識され、プログラム上はセンサやアクチュエータ、計算機と同じ存在になる。

プログラム内において人の役割が重要になってきており、人とセンサ・アクチュエータと計算機は同じ様なコードで扱えるようになることが求められている。同じ様なコードで扱えるようにするためには、センサ・アクチュエータや計算機と同様のインタフェースを人が持ち、同じような動作をする必要がある。例えば、センサからデータを取得する際には、以下のような流れが必要となる。

1. arduino.read_analog(0) のようなメソッドを実行し、センサに対してメッセージを送る
2. その返答としてセンサデータを取得する。

アクチュエータや計算機でも同様のことが言え、命令を送り、その返り値を受け取るようになっている

そこで、「人」に対しても同様の手法を用いることで、センサ・アクチュエータや計算機と同じようなコードで「人」を扱えるようにする。そのために、以下の2つのソフトウェアを実装する。

1. 人への命令記述を可能にするライブラリ
2. 命令を受け取り、返り値を返すアプリケーション

また、主要アプリケーションについて論じた後、分散処理の仕組みの導入についてや、応用例、プロトタイプ実装について述べる。

1.3.1. 人への命令記述を可能にするライブラリ

プログラム内において、人への命令を記述可能にし、命令の返答を待って処理を継続するようなことを可能にするライブラリを開発する。

従来の実世界プログラミング環境においては、センサ・アクチュエータと計算機の関係があり、これらの関係を記述するだけに留まっていた(図1)。本提案によって実現する実世界プログラミング環境においては、センサ・アクチュエータと計算機の関係に、人が入り込み、この3者が相互に通信をしあうことによって、より強力なインタラクションを行えるようになる(図2)。

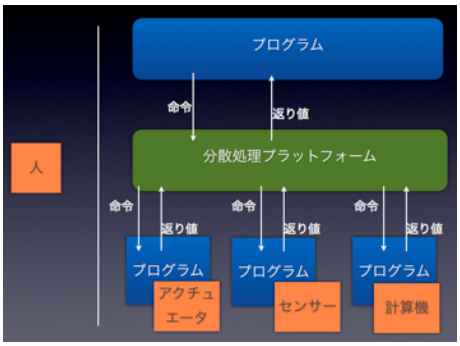

図1. 従来の実世界プログラミング環境

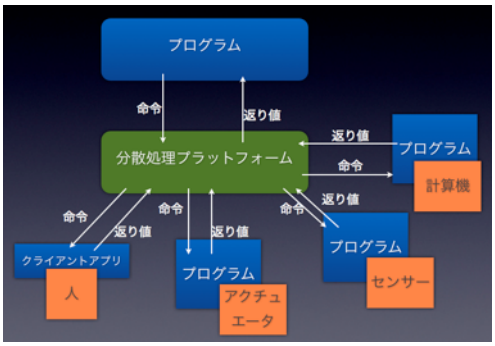

図2. 本提案による新しい実世界プログラミング

そのためにはまず、センサ・アクチュエータと計算機に対してプログラムから命令を送ることと同じように人へ命令を送れるようにしなくてはならない。

```
res = Human.do "研究室にいて、ご飯を食べに行く人の数を教えてください"
if res > 5
  puts "では、ご飯を食べに行きましょう"
else
  puts "人数が少ないので、今日はやめておきましょう"
end
```

図3. 人への命令メソッドの例

具体的には、以下のような、メソッドを実行することによって、「人」に命令を送信し、「人」からの返り値を待って次の処理を行えるようにする(図3)。この場合、Human.do の部分が、人への命令構文になり、引数の部分が命令の内容となる。

1.3.2. 命令を受け取り、返り値を返すアプリケーション

プログラムからの命令を受け取り、人が処理した結果をプログラム側に返すようなクライア

ントアプリケーションを開発する。プログラムから命令を受け取り、処理した結果を返すことは本提案において核となる部分である。例えば、プログラムから「食パンを買ってきてください」といった命令を受け取った場合、「はい」もしくは「いいえ」のような返答をクライアントアプリケーション上からプログラムに送信することができる。

このアプリケーションは、ユーザが常に身に着けていたり、見ているようなデバイス上で動くアプリケーションとして実装する。プログラムからの命令を受け取ったあと、すぐに命令に気づき、その返答ができたほうが良いからだ。そのため、まずはスマートフォンアプリケーションとしての実装を行う。

スマートフォンアプリケーションとして実装することによって、人の状態を判別し、命令を受け取るかどうかといった判断を行うこともできる。例えば、「秋葉原で電子工作の部品を買ってきてください」といった命令が来たとき、自分の居場所が秋葉原から程遠い場所にいた場合、この命令を自分がやることは不適切であり、位置情報的に秋葉原に一番近い、他の人が実行すべきタスクであると言える。本提案では、命令の送信のプラットフォームとして、分散処理の仕組みを導入している。そのため、位置情報などの人のコンテキストを考慮した上でタスクの分散を行える。

1.3.3. 分散処理

上述のような仕組みに分散処理の考えを取り入れることによって、より柔軟に人やセンサ・アクチュエータ、計算機をプログラム内で扱うことができる。状況に応じて、複数の人に命令を分散させることができれば、1人に過大な負荷をかけずに処理することができる。また、人以外にも、センサやアクチュエータ、計算機に対しても同様に処理を分散させることができれば、状況に応じた適切な情報の取得や実世界への干渉を行えるようになる。

分散処理のための仕組みとしては、Lindaを利用する。Lindaとは、並列処理のためのプログラミング言語である。私の所属する研究室では、このLindaをWebから利用可能なLinda-Base[1]という分散処理のためのWebプラットフォームを運用している。並列分散処理のための仕組みをWebから利用できるようにしているため、Webに接続可能なあらゆるデバイスを分散処理の単位として扱うことができる。本提案における、各クライアントへの命令送信は、Linda-Baseを経由して送信される。Linda-Baseはオープンソース・ソフトウェアとして公開されており、LindaをRuby上に実装している。本提案の主な実装である2つのソフトウェアの実装に合わせ、実装の変更などを行う。

1.3.4. 応用例

以下のような応用が考えられる。

- 料理レシピサイトで作る料理を選ぶと、冷蔵庫の中身をセンシングし、足りない食材をリストアップ。出かけている家族に、足りない食材の購入命令を送る
- トイレットペーパーのストックがなくなったことをセンシングし、薬局近くを通った家族

[1] http://linda.masuilab.org/

にトイレットペーパーの購入命令を送る

- コンピュータが処理できないようなエラー値を検出したら、人に通知。人による解釈の後、人からコンピュータにその後の処理についての命令を送る。
- 工場における作業の割り振り・エラー処理などの全プログラム化

1.3.5. プロトタイプ

本提案のプロトタイプとして、Rubyを拡張したBabaScriptと、BabaScriptからの命令を受け取り、返り値を返せるBabaScriptClientを実装した。命令を各BabaScriptClientへ送信するプラットフォームに前述の分散処理プラットフォームLinda-Baseを採用した。

BabaScriptでは、クラス内で定義されていないメソッド全てを人間への命令とすることで、ただ単純に命令として送りたい文章をメソッド風に書くだけで命令を送れるようになっている。

BabaScriptからの命令はLinda-Baseを通して、対象となる人間のクライアントアプリケーションへと送信される。Lindaにはタプルスペースという概念上の共有メモリが存在し、タプルスペースごとに区切ることで、タスクを人であったりグループに通知することができる。例えば、「TakumiBaba」というタプルスペースで命令した場合、TakumiBabaというキーワードを監視しているクライアントに命令を送信することができる。個人名の場合はその個人にのみが監視することで、個人へのタスク通知が可能となる。「sfc」というタプルスペースで命令した場合、sfcの学生全員が「sfc」というタプルスペースを監視するようにしていれば、sfcの学生全員に通知可能であり、分散が可能となる。

BabaScriptから送られた命令を受け取ったBabaScriptClientは、通知機能を用いてユーザに

図4. BabaScript命令返信ビュー

図5. BabaScript受信済み命令リスト

命令が来たことを通知し、ユーザは命令内容を処理した結果をBabaScriptに返すことができ

る。

1.3.6. 今後について

現状のプロトタイプであるBabaScriptでは、各人に対して命令を送信し、その返り値を得ることはできるようになったが、センサー・アクチュエータや計算機とのインタラクションの記述を考慮した実装にはなっていないため、本提案の全てを実装しきれているわけではない。

BabaScriptClientにおいても問題がある。現状の実装においては、ユーザの位置情報であったり状況を考慮したタスクの受信などができない状態にある。タスクを消化可能な人に分散することを実現するためには、より実装を進めていく必要がある。

今後は、プロトタイプの実装から得たフィードバックを元に、さらなる改良を行う。

2. どんな出し方を考えているか

オープンソース・ソフトウェアとしてソースコードを公開し、あらゆる人が利用可能な形にすることで、高度な応用事例の導出を狙う。本提案のような、身近な人を対象として仕事を分散的に振り分けたり、センサー情報に応じて人に命令を送れるような仕組みは未だ普及していない。その要因としては、人をプログラム的に表現できる仕組みが、複雑であったり、そもそも対象としているターゲットが異なっているからだと考えられる。本提案を組み込むことによって、実世界における人とセンサ・アクチュエータや計算機とのインタラクションをプログラム的に記述することは従来に比べ簡略化される。本来、実装に当てられていた時間をアイデア出しに使うこともできるため、より高度な応用事例を見つけ出すことも可能があると考えられる。

また、学会発表やWeb上での情報共有、学内のフォーラムでの発表を通して、より多くの人に本提案について知ってもらうことで、利用を促進する。

3. 斬新さの主張、期待される効果など

プログラム内において人をプロセスのようなモノとして扱い、プログラムから人に命令を送ったり、他の人やセンサー・アクチュエータ、計算機と通信を行えるような仕組みは新しいものである。プログラム内において、人をセンサーやアクチュエータ、計算機と同等に扱えるようになれば、同じようなソースコードで、人を利用してよりパワフルに実世界をコントロールしたり、コンテキストを伴った実世界情報を取得できるようになる。

また、人をプログラム上で表現可能にしたことによって、偉大な先人が作り上げてきたプログラミングパラダイムやテクニックをそのまま人を扱うことに応用可能になる。例えば、Unixにおけるパイプラインは、応用可能な良い例である。センサーデータを取得し、人に渡して処理し、その結果をアクチュエータで表現するといったことはパイプライン的な手法であると言える。パイプライン以外でも、応用可能で有用な仕組みは存在し、先人の知恵を最大限に有効

活用していくことができる。今までの人を扱うような仕組みにおいてはこういった事例は存在しない。

4. 具体的な進め方と予算

4.1. 開発環境

使用する計算機環境

システム開発用: MacBookAir OSX 10.8
　プロセッサ: Core i7　メモリ: 8GB
VPSサーバ: Sakura VPS Ubuntu 12.04
　プロセッサ: 仮想6コア メモリ: 8GB
クライアント用:
　Android: Galaxy Nexus
　iOS: iPod touch

4.2. スケジュール

平日4時間、休日6時間を本提案の実装のために使用する。開発開始から終了まで、おおよそ1200時間程度を実装に費やすことができる。

–2013年12月　プロトタイプVer2
α版リリース、フィードバックを得る
–2014年3月　　プロトタイプVer3
β版リリース、フィードバックを得る
–20146月　プロトタイプVer4
正式版リリース

4.3. 予算内訳

人件費:　1600円/時間 * 1200　1,920,000円
VPSサーバ運用費: 7980 * 9ヶ月 71,820円
デバイス購入費: 39,800円

総額: 2,031,620円

5. 提案者の腕前を証明できるもの

5.1. プロトタイプ実装

本提案のプロトタイプとして実装したandroidアプリケーションを、以下のアドレスにて公開している。https://github.com/TakumiBaba/BabaScript-forAndroid

言語はJavaを利用し、WebSocketライブラリ以外の部分を自分で実装した。

5.2. 婚活SNS DingDongの実装

図6. 婚活SNS DingDong トップページUI

共同研究として、婚活SNS DingDong のサーバサイドからクライアントサイドまでの全てを1人で実装した。婚活SNS DingDongは友人ネットワークを活用した新しい婚活のためのSNSだ。サーバソフトには nginx を、使用言語は Node.js を利用し、データベースにはMongoDBを採用した。DingDongの実装によって手に入れた知識は、本提案においても利用可能な知識である。

6. プロジェクト遂行にあたっての特記事項

未踏IT人材発掘・育成事業への応募に関して、担当教員の許可を得ている。

7. ソフトウェア作成以外の勉強、特技、生活、趣味など

アルバイトではガイアの夜明けという番組の新聞版の調査スタッフをやっている。統計データを探しだし社会現象等との因果関係を見出す、といったことをしている。この仕事が影響し、 統計データを見て、何か面白い特徴を見出すことが好きになっている。

8. 将来のソフトウェア技術について思うこと・期待すること

これからの時代は、誰もがプログラマーとなり、実世界をプログラムするようになっていくと思われる。近い将来、ユビキタスコンピューティングが普及した社会が実現すれば、あらゆる場所にセンサーが埋め込まれ、人々はそのセンサー情報を自由に取得できるようになる。そのセンサー情報を元に、人は生活を少し便利にするようなコードを書くようになるだろう。結果として、ソフトウェア開発の単位はより細かくなり、少しのコードで実世界をダイナミックに動かせるような仕組みが求められるようになるのではないかと考えられる。本職のプログラマーとしては、そんな「生活を少し便利にする」ようなコードを、簡単にかけるようなライブラリを提供していく必要があるのではいかと思う。

また、あらゆるモノをプログラムできるような世界になってほしい、とも考えている。本提案もあらゆるモノをプログラムしていくための1要素であるが、人に限らず、その場の空気や空間を自由にプログラムできるようになれば、プログラマとしてやりがいのある人生を送れるのではないかと思う。