Babascript:

Babascript:

匿名で査読を行うため著者名なし *

Summary.

何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。何かしらの処理を実行するとき、コンピュータが得意なことはコンピュータに、人は人にしかできないことや人がやるべきことに集中すべきだ。

1 はじめに

あらゆるモノ・コトがプログラムで記述されるような未来が近づいているプログラムで処理できることは増え続けている実世界においても、それは言えること実世界志向インタフェースなどの研究は多くの成果を出している Hue などの商品によって、実際に誰でもプログラムできるようになってきているこの流れは変わらず、全世界がプログラム可能になっていくと考えられる。

一方で、コンピュータのみでは処理できないようなこともある人の確認が必要な状況 人の意志の反映が求められる状況人でなくてはいけないような状況こういった問い合わせはプログラムに組み込みづらい結果として、プログラム化・自動実行などが難しくなっている。

コンピュータの方が得意なこともあれば、人のほうが得意なこともある何かしらの処理を実現する上で、コンピュータと人は相補的に動作していくべきである今までは出来なかったようなことの実現

上記の状況を踏まえ、本論文ではプログラムから人へ、処理命令を送れる人力処理組み込み環境 Babascript を提案する。 Babascript 環境では、プログラムというフォーマット上においてコンピュータと人、双方への命令を同様のインタフェースで記述可能になる特殊なプログラミング言語を導入せず、既存の言語上において実現可能

コンピュータはプログラムという手順書を実行する人も、常日頃から手順書に基づいて行動している ex: マニュアル、レシピ手順書のフォーマットが異なるだけで、処理内容が記述された手順書を参照していることには変わらないつまり、手順書のフォーマットは同一にすることも可能である。

* 匿名で査読を行うため所属名なし

2 Babascript

Babascript 環境では、プログラムに人を組み込めるようにするために、関数呼び出しによって人に命令を送れる仕組みを持つ。通常のプログラムでは、関数に引数を与え、実行することで、その関数の名に適した処理が行われた後、返り値として処理結果を得ることが出来る。Babascript では、同様に関数呼び出しし、処理を人に実行させ、返り値を処理結果として返す。プログラム上においては、両者は同じ関数呼び出しだが、その実行主体を人間にすることによって、プログラム内における人力処理の組み込みを実現する。である。

このような仕組みを実現するために、人への命令構文を持った DSL Babascript と、命令に対して返り値を返すためのクライアントライブラリ BabascriptClient と、クライアントライブラリを組み込んだ Web アプリケーションを開発した、また、プラグイン機構によってその機能を拡張可能にした。

2.1 Script

Babascript は、関数呼び出しによって人に対して処理命令を送れるオブジェクト (以下、人オブジェクト) を宣言可能にする DSL だ。人オブジェクトにおいて定義されていない全てのメソッドが人への命令として解釈される。また、命令に対して人からの返り値を得ると、実行メソッドの引数で指定したコールバック関数を実行する。

2.1.1 人オブジェクトの宣言と処理命令構文

図1のようなプログラムによって、人への処理命令を送ることができる。

```
baba = new Baba.Script "baba"
baba.ほげふが (task) ->
  console.log 'hoge'
```

人オブジェクトは宣言時にIDを指定することによって、誰に対して処理命令を配信するかを決定する。実行された処理命令は、指定したIDを監視するクライアントライブラリにのみ配信される。この際、指定したIDを監視するクライアントが複数人、つまり、グループを形成していた場合、人への命令構文を実行時に他のタスクを実行していないクライアントへと順番にタスクが配信される。

人への処理命令構文は、実行されるとメソッド名と引数を元にしたjsonオブジェクトへと変換され、このjsonオブジェクトが処理命令のデータとしてクライアントライブラリに配信される。

人への処理命令構文の第一引数には基本的な処理情報に加えてクライアント側に送信するオプション情報を指定する。第二引数には、人から処理命令に対する返り値が得られた際に実行するコールバック関数を指定する。コールバック関数実行時には、その引数に返り値と関連した情報を格納したオブジェクトが与えられる。

オブジェクトに定義されていない全ての関数は人への処理命令構文として解釈される。この機能はmethodmissingという、定義されていないメソッドが実行された際にその処理を記述しておく仕組みによって実現する。

2.1.2 オプション情報の付加

人への処理命令構文の第一引数に与えるオプション情報には、返り値の型指定などの情報が考えられる。返り値としてあらゆる型を想定したプログラムを記述することは難しい。例えばBoolean型で返してほしい、といった時には図2のようなプログラムが考えられる。

```
baba.ほげふが {format: 'boolean'}, (task) ->
    console.log task
```

他にも、指定したリストの中から値を選択してほしい、といった命令の場合は、そのリストをオプション情報として付加することが考えられる。

特別なオプション情報として、broadcastオプションが存在する。broadcast機能は、指定したIDを監視する全てのクライアントへとタスクを配信し、指定した数の返り値を得られると処理を終了し、コールバック関数を実行するといったものだ。

2.2 Client

Babascript Clientは、Babascriptからの命令受信と処理結果の送信機能を実現するクライアントライブラリだ。命令受信のイベントに対してコールバック関数を指定することで、処理命令を受け取ることができる。基本的な利用方法は、図3に示す。

```
client = new BabascriptClient "baba"
client.on "get_task", (task) ->
    # タスク受信後の動作を記述する
```

クライアントオブジェクトの宣言時、IDを指定することによって、IDに対して処理命令が発行された際にタスクを受信することができる。また、クライアントオブジェクトが持つ “returnValue” メソッドを利用することで、返り値を命令発行元に返すことができる。

このClientライブラリを組み込み、ユーザに提示するためのWebアプリケーションを実装した。タスク情報において型指定をすることによって、ユーザに提示するUIを変化させ、型に合った返り値を選択できるようにしている。例えば、Boolean型を指定していた場合、ユーザにはtrueボタンとfalseボタンが提示され、どちらかを押すと、その結果が返り値としてプログラムに返される。また、String型を指定すれば、文字列の入力フォームとSubmitボタンが表示される。スマートフォンに最適化したWebアプリケーションとして実装しており、様々な場面において利用可能であり、実世界におけるタスクを処理しながらでも十分に利用可能であると言える。

クライアントライブラリは、UIと完全に分離した実装となっており、かつ利用方もシンプルだ。上記のWebアプリケーションだけでなく、新規/既存問わず、様々なアプリケーションに組み込むことができる。

2.3 Plugin

Pluginは、Babascript及びBabascript Clientに組み込むことのできるプラグイン機構だ。Babascript及びBabascriptClientが発行するイベントとデータを受け取ることができる。以下のようなイベントを受け取り、その都度処理を実行させることができる。

- init(プラグイン読み込み時)
- connect(ネットワーク接続時)
- send(タスクの送信時)
- receive(タスクの受信時)

以下のようなプログラムによって、プラグインを読み込むことができる

```
baba = new Babascript("baba")
baba.use(new Logger())
```

具体的には、以下のようなプラグインの実装が挙げられる。

- ログコレクター
- データ同期
- ユーザ管理

2.4 通信と分散処理

script と client のタスク送受信と分散配信のために、node-linda[2] を利用した。Websocket による通信が行われ、リアルタイムにタスクの送受信が行われている。script, client 共に、この node-linda に接続するためのライブラリを組み込んでいる。

script は、プログラム実行時に node-linda サーバに接続し、人への命令構文を実行するごとに、タスクを node-linda サーバに書き込む。書き込み終了後、そのタスクと同じ ID を持った返り値タスクが書き込まれるまで、node-linda サーバの監視を行う。

client は、プログラム起動中は常に node-linda サーバに接続し、指定された ID を監視する。

2.5 実装

上記システムは全て Javascript で実装した。Babascript は Node.js 上で動作し、Babascript Client は Node.js と Web ブラウザ上で動作する。全体図は図 1 のとおりだ。

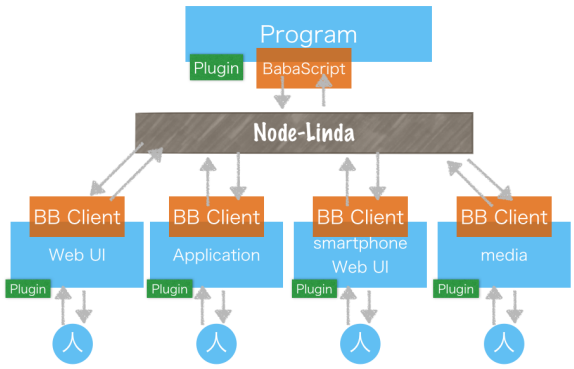

図 1. システム全体図

2.5.1 タスク情報の構成

人への処理命令構文は実行によってタスク情報を生成する。このタスク情報は、以下のような json オブジェクトとなる。

```
task = {
  "name": "baba", # 命令配信先 ID
  "type": "eval", # 配信タイプ
  "key": "パン買ってこい", # 処理命令の内容
  "cid": "'1409227153_0.8690411776769906'",
  # タスク ID
  "option": {
    "format": "boolean",
  }
  # オプション情報
}
```

2.5.2 実行結果の返し方

人への命令構文に対する返り値が、適切に返り値として戻されるようにするために、命令ごとにユニークな ID を生成している。ユニーク ID は、8 桁のランダムな数値と人への命令構文実行時の UnixTime を結合した文字列である。このユニーク ID を伴った返り値が戻ってくるか、指定したタイムアウト時間に到達するか、タスクキャンセルをするまで Babascript は返事を待ち続ける。

2.6 サンプルプログラム

3 応用例

以下のような応用が考えられる。

- 人の行動をプログラミングする
- 実世界をテストする
- あれ

3.1 人の行動をプログラミングする

仕事や役割などは、コンピュータと人の両方の作業で成り立つ人が代替可能になる 人が状態を管理する必要がなくなる

3.2 実世界をテストする

4 議論

4.1 一処理単位としての人

本研究では、人はコンピュータ等と同じ、処理を命令され実行するノードとして扱われると思われる。けど、これは人が人にしかやれないようなことや、人がやるべきことだけに集中できるような環境を構築するためのもの。人から見れば、条件判断を出来るだけプログラムとして記述できるほうが、処理実行もしやすい。

4.2 タスク実行の遅延と実行保障性

Babascript によってタスク実行を依頼しても、人がすぐにタスクを実行し値を返すことを完全に保証することはできない。タスク受信端末を見ていない、受信しても実行できないといった状況の場合、すぐに値を返すことはできない。こういった際、Babascript による処理がボトルネックとなる可能性がある。

また、労働関係にあるなど、タスク実行に強制力がある場合は、タスク実行が確実に行われると考えられるが、強制力がない場合はそもそもタスク受信を無視するといったことも考えられる。タスク実行に強制力がない場合は、金銭などのインセンティブ

を与えるといった手段によって、実行保障性を確保するといったことが考えられる。

4.3 命令内容の粒度

Babascript では、タスクの文面の記述には制限がないため、自由となっている。この文面は、適切な抽象度の文面に設計しなくてはならない。抽象度が高すぎる命令は、あいまいな表記となり、タスク実行者にとって理解しづらい文面となり得る。その結果、想定外の処理が実行され、意図しない結果を招く恐れがある。抽象度が低すぎる命令は、全体の処理内容にもよるが、プログラム自体が冗長となり得る。プログラムとタスク実行者の間のやりとりが増え、通信や待機時間などがボトルネックとなる可能性がある。また、タスク実行者にとっても、やりとりが増えることで負担増になると考えられる。

4.4 同時の複数命令

複数のプログラムから同時に一人のタスク実行者へとタスクが配信される可能性がある。この際、異なるコンテキストにある命令が交互に配信され、タスク実行に大きな障害をもたらす可能性がある。例えば、料理プログラムと掃除プログラムが同時に実行された場合、鍋で煮ている途中で「洗剤を投入しろ」などといった命令が配信されることが考えられる。

この問題は、全ての Babascript プログラム中において、一人のタスク実行者は一つのプログラムからのみ、連続してタスクを受信できるような仕組みを用意することによって、解決可能であると考えられる。また、応用アプリケーションでの実装になるが、コンテキストを明示し、どの処理系におけるタスクなのかをタスク実行者に示すといった手段によっても解決可能である。

5 関連研究

計算機では処理が難しいようなタスクを解決するために、人を計算資源として利用する手法はヒューマンコンピュテーション [8] と呼ばれ、様々な研究が行われている。インターネットを介して不特定多数の群衆にタスクを実行させるクラウドソーシングと組み合わせた研究事例も多く存在する。クラウドソーシングのプラットフォームとしては、Amazon Mechanical Turk[1] が存在する。Barowy らは、CrowdProgramming という概念を提唱し、プログラミング言語内においてクラウドソーシングによる計算とコンピュータによる計算の統合を実現した [3]。Franklin らは、機械だけでは答えられないような DB へのクエリに対する応答を、クラウドソーシングを用いることで返答可能にする CrowdDB を提案している [6]。Morishima らは、人をデータソースとしてプログラムの中で利用する手法を提案している [7]。jabberwocky crowdforge これらの研究では、人を計算資源やデータソースとしてシステムに組み込むことを狙っている。本研究では、計算資源やデータソースに限らず、実世界への干渉等も対象としており、本研究はより汎用的な枠組みとなっている。また、いずれもクラウドソーシングの利用を前提としているが、本研究は不特定多数の群衆を対象としたものではない。家族であったり、職場の人であったりといった、特定可能な人を対象としている。

ユビキタスコンピューティングの研究分野においては、Human as Sensor という概念も提唱されている。PRISM は、スマートフォンを利用したセンシングプラットフォームだ [5]。Liu らは、ソーシャルメディア上の人をセンサーとして扱った Q&A サービス MoboQ を提案し、その検証を行った。Human as Sensor に類する研究では、人をセンサーとして扱うことを対象としているが、本研究ではセンサーのみを対象としていない。

Cheng らは、人をモーションプラットフォームにおけるモーターやメカニカル機構の代替として利用した Haptic Turk を提案している [4]。Haptic Turk はゲームでの利用に特化したものだ。本研究は、使用用途を限らない汎用的な仕組みとなっている。

加藤らは、人とロボット間でのタスク共有システム Sharedo を提案した [9]。人とロボットのタスク実行における協調は、本研究の主眼である「何かの処理を実現するとき、人とコンピュータは相補的に動作するべき」という考えと大きく類似している。

6 おわりに

本論文では、人への命令構文をプログラムに付与可能なプログラミング環境 Babascript を提案した。Babascript 環境においては、プログラム上において人は、コンピュータと同じ処理ノードとして存在し、関数実行によって処理内容を受け取り、実行・値を返す存在になれる。これによって、プログラマブルになりつつある世界において人自身もその一部になることが可能となる。

また、Babascript 環境によって実現する応用例を示すことによって、人をプログラマブルにした際のメリットを示した。

今後は、議論で述べた Babascript の問題点などを改善していく。

参考文献

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/.

[2] Linda implementation on Socket.IO. https://github.com/node-linda/linda.

[3] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor. AutoMan: A Platform for Integrating Human-based and Digital Computation.

In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pp. 639–654, New York, NY, USA, 2012. ACM.

[4] L.-P. Cheng, P. Lühne, P. Lopes, C. Sterz, and P. Baudisch. Haptic Turk: A Motion Platform Based on People. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 3463–3472, New York, NY, USA, 2014. ACM.

[5] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: Platform for Remote Sensing Using Smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pp. 63–76, New York, NY, USA, 2010. ACM.

[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: Answering Queries with Crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pp. 61–72, New York, NY, USA, 2011. ACM.

[7] A. Morishima, N. Shinagawa, T. Mitsuishi, H. Aoki, and S. Fukusumi. CyLog/Crowd4U: A Declarative Platform for Complex Data-centric Crowdsourcing. *Proc. VLDB Endow.*, 5(12):1918–1921, Aug. 2012.

[8] L. Von Ahn. *Human Computation*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3205378.

[9] 淳 加藤, 大介 坂本, 健夫 五十嵐. Sharedo: To-do リストによる人-ロボット間のタスク共有. WISS '11: 第19回インタラクティブシステムとソフトウェアに関するワークショップ, pp. 102–107, 2011.